Chapter 6

Microsoft® DirectX™ 3 Software Development Kit

DirectInput

DirectInput

CHAPTER 6

New Information for DirectX 3

Microsoft® DirectInput™ now includes support for mouse and keyboard input devices, as well as for joysticks. The API for the mouse and keyboard use COM objects and interfaces. The mouse and keyboard support is described in a Word document that is part of the DirectX™ 3 SDK but is not part of this document set.

Joysticks are currently supported by the API documented in the rest of this chapter.

Overview

Introduction to Joysticks

The DirectInput application programming interface (API) provides fast and consistent access to analog and digital joysticks. The DirectInput API maintains consistency with the joystick APIs of the Microsoft Win32® Software Development Kit (SDK), but has improved responsiveness and reliability by changing the device driver model. DirectInput device drivers use the registry to store settings for standard and OEM-supplied joysticks, and calibration information for previously configured joysticks.

This section describes DirectInput functions, messages, and structures that support joysticks, while it updates the Win32 joystick API documentation. The DirectInput APIs also apply to other ancillary input devices that track positions within an absolute coordinate system, such as touch screens, digitizing tablets, and light pens. Extended capabilities also provide support for rudder pedals, flight yokes, virtual-reality headgear, and other devices. Each device can use up to six axes of movement, a point-of-view hat, and 32 buttons.

You can design your application to use DirectInput functions to determine the capabilities of the joysticks and joystick drivers. Also, your application can query a joystick to process its position and button information.

Joystick Capabilities

DirectInput services are loaded when the operating system is started. DirectInput supports analog and digital joysticks. Analog joysticks require real-time responsiveness and place a higher burden on the system than digital joysticks. DirectInput services can simultaneously monitor analog joysticks in a number of different configurations. These configurations range from two analog joysticks that track up to four axes of movement and use up to four buttons to four analog joysticks that track two axes of movement and use up to four buttons. In contrast, DirectInput services can simultaneously monitor up to 16 digital joysticks, each with up to six axes of movement and up to 32 buttons.

Note

Each axis of movement that a joystick tracks has a range of motion. The range of motion is the distance a joystick handle can move from its neutral, or resting, position to the farthest points from that resting position.

The joystick driver can support up to 16 minidrivers, with each minidriver supporting one joystick. You can retrieve the number of joysticks supported by a joystick driver by using the **joyGetNumDevs** function. This function returns an unsigned integer specifying the number of joysticks that the driver can support, or 0 if there is no joystick support.

Your application can determine if a joystick is attached to the computer by using the **joyGetPosEx** function. This function returns JOYERR_NOERROR if the specified device is attached, or JOYERR_UNPLUGGED otherwise.

Each joystick has several capabilities that are available to your application. You can retrieve the capabilities of a joystick by using the **joyGetDevCaps** function. This function fills a **JOYCAPS** structure with joystick capabilities, such as valid axes of movement for the joystick, minimum and maximum values for its coordinate system, and the number of buttons on the joystick.

The return value of **joyGetNumDevs** does not indicate the number of joysticks attached to the system, but the number of joysticks that the system supports.

Joystick Calibration and Testing

Microsoft Windows® 95 provides a joystick application in the Control Panel to calibrate and test joysticks, including range of motion and buttons. This application lets the user choose from a list of joysticks, including the following:

- Generic joystick configurations
- OEM joysticks
- Custom joysticks

The application allows the calibration of up to six axes of motion, 32 buttons, and a point-of-view hat for each joystick. Calibration information is stored in the registry, so the user can switch from one joystick to another without recalibrating each one every time it is used. Once the user has calibrated or selected a new joystick, the calibration application updates the registry with the selected joystick and calibration information, and notifies the joystick driver accordingly.

Additionally, this application can notify a particular joystick of changes to the registry that affect the joystick, by using the **joyConfigChanged** function.

Joystick Position

You can query a joystick for position and button information by using the **joyGetPosEx** function. This function reports position information returned by the older joystick functions of the Win32 API, including position coordinates for the x-, y-, and z-axes. It also reports state information for up to four buttons. The **joyGetPosEx** function also provides access to the following information:

- Status of fourth, fifth, and sixth axes (r, u, and v)
- Rudder information
- Point-of-view hat
- State information for up to 32 buttons
- Uncalibrated (raw) joystick data
- Scaled data for a defined range of values
- Centered scaled data
- Scaled data that includes a dead zone around the neutral joystick position

Reference

Joystick Groups

This section describes the functions and structures associated with joysticks. The elements are grouped as follows:

Device capabilities	**JOYCAPS**
	joyConfigChanged
	joyGetDevCaps
	joyGetNumDevs
Querying a joystick	**joyGetPosEx**
	JOYINFOEX

Functions

joyConfigChanged

```
MMRESULT joyConfigChanged(DWORD dwFlags);
```

Notifies the joystick driver that the registry contains new joystick settings.

- Returns JOYERR_NOERROR if successful, or one of the following error values otherwise:

  JOYERR_NOCANDO

  JOYERR_REGISTRYNOTVALI D

dwFlags
Reserved, must be set to 0.

The joystick calibration property sheet in the Control Panel calls this function when the user calibrates or recalibrates a joystick, or when a different joystick device is selected.

If your application is designed to customize joystick performance, such as an OEM joystick calibration application, you can use this function to notify the joystick driver that the JOYSTICK_USER values in the registry for the currently selected joystick have changed. The JOYSTICK_USER values are located in the HKEY_LOCAL_MACHINE hive of the registry.

joyGetDevCaps

```
MMRESULT joyGetDevCaps(UINT uJoyID, LPJOYCAPS pjc,
    UINT cbjc);
```

Queries a joystick to determine its capabilities.

- Returns JOYERR_NOERROR if successful, or one of the following error values otherwise:

  MMSYSERR_INVALPARAM

  MMSYSERR_NODRIVER

uJoyID
Identifier of the joystick, either JOYSTICKID1 or JOYSTICKID2, to be queried.

pjc
Address for a **JOYCAPS** structure that will contain the joystick's capabilities.

cbjc
Size of the **JOYCAPS** structure, in bytes.

You can use the **joyGetNumDevs** function to determine the number of joystick devices the driver supports.

See also **JOYCAPS**, **joyGetNumDevs**

joyGetNumDevs

```
UINT joyGetNumDevs(VOID);
```

Queries the joystick driver for the number of joysticks it supports.

- Returns the number of joysticks the driver supports, or 0 if no driver is present.

The **joyGetPosEx** function is used to determine whether a given joystick is physically attached to the computer.

See also **joyGetDevCaps**

joyGetPosEx

```
MMRESULT joyGetPosEx(UINT uJoyID, LPJOYINFOEX pji);
```

Queries a joystick for its position and button status.

- Returns JOYERR_NOERROR if successful, or one of the following error values otherwise:

  JOYERR_UNPLUGGED

  MMSYSERR_BADDEVICEID

  MMSYSERR_INVALPARAM

  MMSYSERR_NODRIVER

uJoyID
Identifier of the joystick to be queried.

pji
Address for a **JOYINFOEX** structure that contains extended position information and button status of the joystick.

Before calling this function, your application must identify the items to query by setting one or more flags in the **dwFlags** member of the **JOYINFOEX** structure.

This function provides access to extended devices, such as rudder pedals, point-of-view hats, devices with a large number of buttons, and coordinate systems using up to six axes.

See also **JOYINFOEX**

Structures

JOYCAPS

```
typedef struct {
    WORD wMid;
    WORD wPid;
    CHAR szPname[MAXPNAMELEN];
```

```
    UINT wXmin;
    UINT wXmax;
    UINT wYmin;
    UINT wYmax;
    UINT wZmin;
    UINT wZmax;
    UINT wNumButtons;
    UINT wPeriodMin;
    UINT wPeriodMax;
\\  The following members are not in previous versions
\\  of Windows.
    UINT wRmin;
    UINT wRmax;
    UINT wUmin;
    UINT wUmax;
    UINT wVmin;
    UINT wVmax;
    UINT wCaps;
    UINT wMaxAxes;
    UINT wNumAxes;
    UINT wMaxButtons;
    CHAR szRegKey[MAXPNAMELEN];
    CHAR szOEMVxD[MAXOEMVXD];
} JOYCAPS;
```

Contains information about the specified joystick's capabilities.

wMid
Manufacturer identifier.

wPid
Product identifier.

szPname
Null-terminated string that contains the joystick product name.

wXmin and **wXmax**
Minimum and maximum x-coordinate values.

wYmin and **wYmax**
Minimum and maximum y-coordinate values.

wZmin and **wZmax**
Minimum and maximum z-coordinate values.

wNumButtons
Number of joystick buttons.

wPeriodMin and **wPeriodMax**
Minimum and maximum polling frequencies supported once an application has captured a joystick.

wRmin and **wRmax**
Minimum and maximum rudder values. The rudder is the fourth axis of movement.

wUmin and **wUmax**
Minimum and maximum u-coordinate (fifth axis) values.

wVmin and **wVmax**
Minimum and maximum v-coordinate (sixth axis) values.

wCaps
Joystick capabilities. The following flags define individual capabilities that a joystick might have:

JOYCAPS_HASPOV
The joystick has point-of-view information.

JOYCAPS_HASR
The joystick has rudder (fourth axis) information.

JOYCAPS_HASU
The joystick has u-coordinate (fifth axis) information.

JOYCAPS_HASV
The joystick has v-coordinate (sixth axis) information.

JOYCAPS_HASZ
The joystick has z-coordinate information.

JOYCAPS_POV4DIR
The joystick point-of-view supports discrete values (centered, forward, backward, left, and right).

JOYCAPS_POVCTS
The joystick point-of-view supports continuous degree bearings.

wMaxAxes
Maximum number of axes the joystick supports.

wNumAxes
Number of axes currently in use by the joystick.

wMaxButtons
Maximum number of buttons the joystick supports.

szRegKey
Null-terminated string that contains the registry key for the joystick.

szOEMVxD
Null-terminated string that identifies the joystick driver OEM.

See also **joyGetDevCaps**

JOYINFOEX

```
typedef struct joyinfoex_tag {
    DWORD dwSize;
    DWORD dwFlags;
    DWORD dwXpos;
    DWORD dwYpos;
    DWORD dwZpos;
    DWORD dwRpos;
    DWORD dwUpos;
    DWORD dwVpos;
    DWORD dwButtons;
    DWORD dwButtonNumber;
    DWORD dwPOV;
    DWORD dwReserved1;
    DWORD dwReserved2;
} JOYINFOEX;
```

Contains extended information about the joystick position, the point-of-view position, and the button state.

dwSize
Size of this structure, in bytes.

dwFlags
Flags that indicate if information returned in this structure is valid. Members that do not contain valid information are set to 0. The following flags are defined:

JOY_RETURNALL

Equivalent to setting all the JOY_RETURN values except JOY_RETURNRAWDATA.

JOY_RETURNBUTTONS

The **dwButtons** member contains valid information about the state of each joystick button.

JOY_RETURNCENTERED

Centers the joystick neutral position to the middle value of each axis of movement.

JOY_RETURNPOV

The **dwPOV** member contains valid information about the point-of-view control, which is expressed in discrete units.

JOY_RETURNPOVCTS

The **dwPOV** member contains valid information about the point-of-view control expressed in continuous, one-hundredth of a degree units.

JOY_RETURNR

The **dwRpos** member contains valid rudder pedal information. This indicates the existence of a fourth axis.

JOY_RETURNRAWDATA

Indicates that uncalibrated joystick readings are stored in this structure.

JOY_RETURNU

The **dwUpos** member contains valid data for a fifth axis of the joystick, if such an axis is available. 0 is returned otherwise.

JOY_RETURNV

The **dwVpos** member contains valid data for a sixth axis of the joystick, if such an axis is available. 0 is returned otherwise.

JOY_RETURNX

The **dwXpos** member contains valid data for the x-coordinate of the joystick.

JOY_RETURNY

The **dwYpos** member contains valid data for the y-coordinate of the joystick.

JOY_RETURNZ

The **dwZpos** member contains valid data for the z-coordinate of the joystick.

JOY_USEDEADZONE

Expands the range for the neutral position of the joystick and calls this range the dead zone. The joystick driver returns a constant value for all positions in the dead zone.

The following flags provide data to calibrate a joystick and are intended for custom calibration applications:

JOY_CAL_READ3

Reads the x-, y-, and z-coordinates, and then stores the raw values in the **dwXpos**, **dwYpos**, and **dwZpos** members, respectively.

JOY_CAL_READ4

Reads the rudder information and the x-, y-, and z-coordinates, and then stores the raw values in the **dwRpos**, **dwXpos**, **dwYpos**, and **dwZpos** members, respectively.

JOY_CAL_READ5

Reads the rudder information and the x-, y-, z-, and u-coordinates, and then stores the raw values in the **dwRpos**, **dwXpos**, **dwYpos**, **dwZpos**, and **dwUpos** members, respectively.

JOY_CAL_READ6

Reads the raw v-axis data if a joystick minidriver is present to provide the information. Returns 0 otherwise.

JOY_CAL_READALWAYS

Reads the joystick port even if the driver does not detect a device.

JOY_CAL_READRONLY

Reads the rudder information if a joystick minidriver is present to provide the data, and then stores the raw value in the **dwRpos** member. Returns 0 otherwise.

JOY_CAL_READUONLY

Reads the u-coordinate if a joystick minidriver is present to provide the data, and then stores the raw value in the **dwUpos** member. Returns 0 otherwise.

JOY_CAL_READVONLY

Reads the v-coordinate if a joystick minidriver is present to provide the data, and then stores the raw value in the **dwVpos** member. Returns 0 otherwise.

JOY_CAL_READXONLY

Reads the x-coordinate and stores the raw value in the **dwXpos** member.

JOY_CAL_READXYONLY

Reads the x- and y-coordinates and stores the raw values in the **dwXpos** and **dwYpos** members, respectively.

JOY_CAL_READYONLY

Reads the y-coordinate and stores the raw value in the **dwYpos** member.

JOY_CAL_READZONLY

Reads the z-coordinate and stores the raw value in the **dwZpos** member.

dwXpos, **dwYpos**, and **dwZpos**
Current x-coordinate, y-coordinate, and z-coordinate positions, respectively.

dwRpos
Current position of the rudder or fourth joystick axis.

dwUpos and **dwVpos**
Current fifth axis and sixth axis positions, respectively.

dwButtons
Current state of the 32 joystick buttons. The value of this member can be set to any combination of JOY_BUTTON*n* flags, where *n* is a value ranging from 1 to 32. Each value corresponds to the button that is pressed.

dwButtonNumber
Current button number that is pressed.

dwPOV
Current position of the point-of-view control. Values for this member range from 0 to 35,900. These values represent each view's angle, in degrees, multiplied by 100.

dwReserved1 and **dwReserved2**
Reserved, do not use.

The value of the **dwSize** member is also used to identify the version number for the structure when it is passed to the **joyGetPosEx** function.

Most devices with a point-of-view control have only five positions. When the JOY_RETURNPOV flag is set, these positions are reported by using the following JOY_POV constants.

Point-of-view positions	Description
JOY_POVBACKWARD	Point-of-view hat is pressed backward. The value 18,000 represents an orientation of 180.00 degrees.
JOY_POVCENTERED	Point-of-view hat is in the neutral position. The value -1 means the point-of-view hat has no angle to report.
JOY_POVFORWARD	Point-of-view hat is pressed forward. The value 0 represents an orientation of 0.00 degrees.
JOY_POVLEFT	Point-of-view hat is being pressed to the left. The value 27,000 represents an orientation of 270.00 degrees.
JOY_POVRIGHT	Point-of-view hat is pressed to the right. The value 9,000 represents an orientation of 90.00 degrees.

The default Windows 95 joystick driver currently supports these five discrete directions. If your application can accept only the defined point-of-view values, it must use the JOY_RETURNPOV flag. If your application can accept other degree readings, it should use the JOY_RETURNPOVCTS flag to obtain continuous data if it is available. The JOY_RETURNPOVCTS flag also supports the JOY_POV constants used with the JOY_RETURNPOV flag.

See also **joyGetPosEx**

Return Values

Errors are represented by negative values and cannot be combined. This table lists the failures that can be returned by all DirectInput functions. For a list of the error codes each function can return, see the individual function descriptions.

JOYERR_NOCANDO

The joystick driver cannot update the device information from the registry.

JOYERR_NOERROR

The request completed successfully.

JOYERR_REGISTRYNOTVALID

One or more registry joystick entries contain invalid data.

JOYERR_UNPLUGGED

The specified joystick is not connected to the computer.

MMSYSERR_BADDEVICEID

The specified joystick identifier is invalid.

MMSYSERR_INVALPARAM

An invalid parameter was passed.

MMSYSERR_NODRIVER

The joystick driver is not present.