AUF CD

Die Quelltexte sowie die fertig übersetzten Routinen finden Sie auf unserer Heft-CD im Verzeichnis *Praxis/PC Underground*.

DirectX 7 – Eingabegeräte ansteuern

Auf Knopfdruck

Mit DirectInput sind Sie Herr der Eingabesysteme. Gleichgültig, ob Sie Maus, Tastatur, Joystick, Game Controller oder Force-Feedback-Geräte ansteuern. Lesen Sie, wie Sie Ein- und Ausgabedaten verarbeiten.

C. DACHSBACHER/
O. KÄFERSTEIN

Wenn Sie ein Programm entwickeln, das mit Maus und Tastatur auskommt, können Sie auf DirectInput (Application Programming Interface für komplexe Ein-/Ausgabegeräte) verzichten. Dann können Sie alle Eingaben des Benutzers in der Windows-Message-Schleife über Schlüsselwörter wie *WM_CHAR*, *WM_MOUSEMOVE* abfragen und behandeln. DirectInput verwenden Sie, wenn Sie in extrem kurzen Zeitintervallen Daten des Eingabegeräts benötigen oder spezielle Eingabegeräte wie Force-Feedback-Joysticks, Lenkräder oder Ähnliches einsetzen wollen.

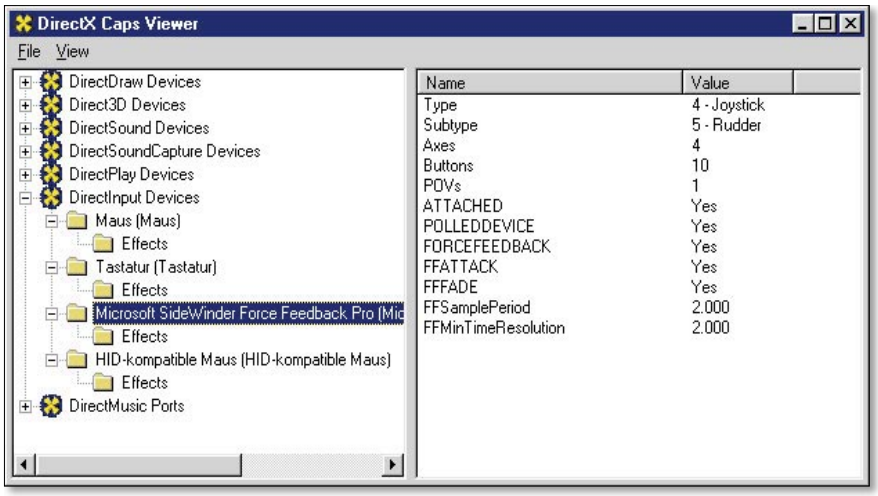
Erst wenn Sie DirectInput initialisiert haben, können Sie die Eingabegeräte ansprechen. Mit dieser Voraussetzung programmieren Sie so, dass Sie die Handlungen Ihrer Programmnutzer verarbeiten. Force-Feedback-Geräte (Eingabegeräte mit Datenströmen vom und zum PC) erhalten via DirectInput ihre Wirksamkeit.

DirectInput arbeitet direkt mit den Gerätetreibern zusammen, Maus- und Tastatur-Nachrichten (elektronische Schaltimpulse) werden unterdrückt oder ignoriert. Ignoriert bedeutet, sie sind für das Input-System uninteressant und werden weitergeleitet. Unterdrückt bedeutet, dass DirectInput die Daten von den Eingabegeräten verwendet. Dabei werden die Nachrichten, die normalerweise an Windows oder Fenster auf dem Bildschirm ausgegeben würden, nicht weitergeleitet. Wenn Sie also die Maus mit DirectInput im *Exclusive Mode* verwenden, kann Windows nicht einmal den Standard-Mauscursor darstellen, da es keine Nachrichten über den Zustand der Mausknöpfe oder Bewegungen bekommt.

Wegen des engen Zusammenspiels mit den Gerätetreibern ignoriert DirectInput Einstellungen, die Sie in der Systemsteuerung für Maus oder Tastatur vornehmen. Bei der Tastatur kennt DirectInput auch keine Tastenwiederholung. Es wird nur für jede Taste unterschieden, ob sie gerade gedrückt oder losgelassen ist.

Auch bei der Maus kennt DirectInput keine Einstellungen der Systemsteuerung wie Beschleunigung des Cursors oder vertauschte Knöpfe. Daher gehen alle Daten am Kontrollsystem von Windows (dem Subsystem) vorbei, das sonst für die Interpretation der Mausdaten zuständig wäre.

Einstellungen, die Sie direkt in den Treibern für die Eingabegeräte vornehmen, werden berücksichtigt. Wenn Sie eine Drei-Tasten-Maus besitzen und den Treiber anweisen, einen Klick mit der mittleren Maustaste als Doppelklick zu verwenden, meldet DirectInput einen Klick mit dieser Taste als zwei Klicks mit der ersten Maustaste.



HIER SEHEN SIE DIE Informationen von DirectX über einen Force-Feedback-Joystick.

TIPP **Keine Regel ohne Ausnahme: Bei Joysticks verwendet DirectInput die Kalibrierungsinformationen, die der Benutzer in der Systemsteuerung eingestellt hat.**

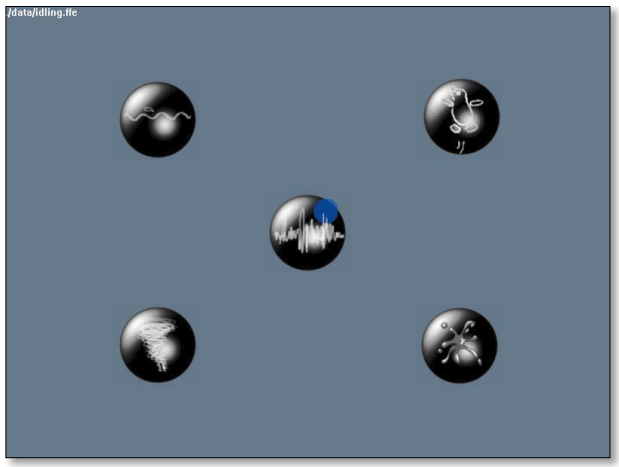
Bei den Schnittstellen verschiedener Geräteeingaben darf das Human Interface Device (HID) nicht fehlen. Dabei handelt es sich um eine Klasse innerhalb der Universal-Serial-Bus-Standards (USB), welche USB-Eingabegeräte ansteuert und auswertet. DirectInput unterstützt alle Geräte, die nach dem HID-Standard funktionieren. Im Gegensatz zu traditionellen Eingabegeräten können HID-Geräte zusätzlich Daten ausgeben. So können Sie zum Beispiel (wie bei Assembler-Programmen unter MS-DOS) Keyboard-LEDs an- oder ausschalten.

Wenn Sie mit DirectInput ein HID-Gerät auswählen, können Sie dessen Art und Fähigkeiten auslesen, weil vordefinierte Codes Sie darüber informieren. Über den HID-Standard finden Sie weitere Informationen unter der Adresse www.usb.org.

Programm mit DirectInput

Wie bei anderen Komponenten des DirectX-Systems gibt es auch für DirectInput ein DirectInput-Objekt. Dieses unterstützt die IDirectInput7-COM-Schnittstelle, die über die Methode *CreateDeviceEx* einen Zeiger auf *IDirectInputDevice7*-Geräte liefert. Das sind Eingabegeräte die eigene Features mitbringen. Dazu zählen Knöpfe und Tasten mit Reglern, die eine Position im Raum repräsentieren, Mausrädchen oder Joysticks.

Um DirectInput-Eingaben zu verarbeiten greifen Sie zunächst über die *DirectInputCreateEx*-Funktion auf das *IDirectInput7*-Interface zu.



MIT DIESEN SCHALTKNÖPFEN im Beispielprogramm sehen Sie Force-Feedback-Geräte in Aktion.

Die mit einem Unterstrich gekennzeichneten Variablen nehmen Sie als global an. Dabei handelt es sich um *Instance Variables*, die Ihren Programmen ebenso eigen sind wie den Variablen der angeforderten Eingabegeräte.

```
// DirectInput Interface
  LPDIRECTINPUT7 _di;
  HRESULT hr;
  hr = DirectInputCreateEx
 ( _hinst, DIRECTINPUT_VERSION,
IID_IDirectInput7, (void**)
 &_di, NULL );
  if( FAILED( hr ) )
    return „geht nicht“;
```

Nachdem Sie das *IDirectInput7*-Interface initialisiert haben, können Sie verschiedene *IDirectInputDevice7*-Objekte erzeugen lassen. Das bedeutet, dass Sie Zugriffe auf einzelne Eingabegeräte anfordern können. Die wichtigsten sind *GUID* (Global Unique IDentifier = global eindeutige Nummer zur Identifizierung):

```
GUID_SysMouse,
GUID_SysKeyboard
GUID_Joystick
```

Unter Windows stehen Ihnen eine Maus und eine Tastatur zur Verfügung. Ohne angeschlossene Tastatur bootet ein PC nicht, womit auch der Start von Windows entfällt. Startet Windows, bemängelt es das Fehlen einer Maus.

Den abgezogenen Joystick-Anschluss bemängelt Windows hingegen nicht. Deshalb ist der Eintrag *GUID_Joystick* keine *Instance GUID*, sondern eine *Produkt GUID*. Wenn Sie einen Joystick mit *CreateDeviceEx* anfordern, müssen Sie berücksichtigen, dass das Kontrollsystem von Windows keine eventuellen Fehler meldet.

Wenn an einem PC mehrere Mäuse oder Tastaturen angeschlossen sind, werden die Geräte zusammengefasst. Sie können dann jede Maus oder Tastatur benutzen. DirectInput kennt noch weitere GUIDs:

```
GUID_SysMouseEm,
GUID_SysMouseEm2,
GUID_SysKeyboardEm,
GUID_SysKeyboardEm2
```

Diese Einträge dienen nur Testzwecken, in unserem Beispiel setzen Sie sie nicht ein. Der Unterschied zu den in unserem Programm aufgerufenen Maus- und Tastatur-GUIDs ist, dass bei den Test-GUIDs alle Eingaben über eine zusätzliche Emulationsschicht laufen.

Wenn Sie nicht wissen, welche Geräte sich an einem Rechner befinden oder welche installiert sind, können Sie nach ihnen suchen. Dabei hilft Ihnen die Methode

```
IDirectInput7::EnumDevices
```

Alle unterstützten Geräte dieser Art finden Sie in der Textbox auf S. 235.

Geben Sie die folgenden Zeilen Code ein, um auf Maus oder Tastatur zuzugreifen:

```
LPDIRECTINPUTDEVICE7 _mouse;
  HRESULT hr;
  hr = _di->CreateDeviceEx
(GUID_SysMouse,
IID_IDirectInputDevice7,
(void**)&_mouse, NULL);
  if( FAILED( hr ) )
    return „geht nicht“

  LPDIRECTINPUTDEVICE7 _key-
board;
  hr = _di->CreateDeviceEx
(GUID_SysKeyboard,
IID_IDirectInputDevice7,
(void**)&_keyboard, NULL);
  if( FAILED( hr ) )
    return „geht nicht“
```

Joystick-Geräte, die Sie nicht an jedem Rechner voraussetzen können, müssen Sie mit der Syntax

```
IDirectInput7::EnumDevices
```

suchen. Wie bei allen Windows-*Enum*-Methoden schreiben Sie eine Callback-Prozedur, die Sie mit der Syntax

```
IDirectInput7::EnumDevices
```

aufrufen. Als Parameter erhält die Callback-Prozedur bei jedem Aufruf eine Beschreibung von einem gefundenen Eingabegerät, und Sie können sich eines davon – sofern vorhanden – aussuchen.

Mit dem ersten Parameter von *IDirectInput7::EnumDevices* geben Sie an, welche Art von Eingabegerät Sie suchen: Joystick, Maus, Tastatur oder HID-Gerät. Sie können die Suche einschränken, indem Sie die Flags (der letzte Parameter bei *IDirectInput7::EnumDevices*) entsprechend setzen. Sie können ausschließlich nach Force-Feedback-Geräten suchen oder nach Geräten, die gerade angeschlossen und nicht nur installiert sind.

Das folgende Codebeispiel demonstriert, wie Sie einen angeschlossenen Joystick suchen und finden:

```
  LPDIRECTINPUTDEVICE7 _joy;

  // Die Callback-Prozedur
  BOOL CALLBACK EnumCallback
(LPCDIDEVICEINSTANCE
pdidInstance, LPVOID pvRef )
  {
    // Interface holt Joystick
HRESULT hr = _di->CreateDeviceEx
( pdidInstance->guidInstance,
IID_IDirectInputDevice7,pvRef),
NULL );

    if( FAILED( hr ) )
      return DIENUM_CONTINUE;

    // Enumeration beenden
    return DIENUM_STOP;
  }

  ...

//Suche ANGESCHLOSSENEN Joystick
  hr = _di->EnumDevices
```

DATENFORMATE VON UND FÜR EINGABEGERÄTE

Datenformat	Struktur mit Zustand
c_dfDIMouse	DIMOUSESTATE
c_dfDIMouse2	DIMOUSESTATE2
c_dfDIKeyboard	array of 256 bytes
c_dfDIJoystick	DIJOYSTATE
c_dfDIJoystick2	DIJOYSTATE2

DIE *DIJOYSTATE2*-STRUKTUR

Syntax	Information
`IX, IY, IZ`	Joystick *X*, *Y* und *Z* Achse (meistens links/rechts, vorwärts/rückwärts und Throttle-Funktion)
`lRx, lRy, lRz`	Rotation der Achsen
`rglSlider[2]`	Zwei zusätzliche Achsen (Slider)
`rgdwPOV[4]`	Positionen von Richtungs-Controllern
`rgbButtons[128]`	Array mit den Zuständen der Knöpfe
`lVX, lVY, lVZ`	Geschwindigkeit der Achsen
`lVRx, lVRy, lVRz`	Winkelgeschwindigkeit der Achsen
`rglVSlider[2]`	Geschwindigkeit zusätzlicher Achsen
`lAX, lAY, lAZ`	Beschleunigung entlang der Achsen
`lARx, lARy, lARz`	Winkelbeschleunigung entlang der Achsen
`rglASlider[2]`	Beschleunigung an zusätzlichen Achsen
`lFX, lFY, lFZ`	Kraft auf den Achsen
`lFRx, lFRy, lFRz`	Drehmoment an den Achsen
`rglFSlider[2]`	Kräfte an zusätzlichen Achsen

```
( DIDEVTYPE_JOYSTICK,
&EnumCallback, (void**)&_joy,
DIEDFL_ATTACHEDONLY );
```

Das *DIEDFL_ATTACHEDONLY*-Flag setzt voraus, dass der Joystick korrekt angeschlossen ist. Wenn Sie zusätzlich Force-Feedback-Fähigkeit verlangen, verwenden Sie folgenden Aufruf:

```
hr = _di->EnumDevices
(DIDEVTYPE_JOYSTICK,
&EnumCallback, (void**)&_joy,
DIEDFL_ATTACHEDONLY |
DIEDFL_FORCEFEEDBACK );
```

Mäuse mit Force-Feedback-Eigenschaften rufen Sie so ab:

```
  LPDIRECTINPUTDEVICE7 _mausFF;
  hr = _di->EnumDevices
(DIDEVTYPE_MOUSE, &EnumCallback,
(void**)&_mausFF,
DIEDFL_FORCEFEEDBACK );
  if( FAILED( hr ) )
    return „geht nicht“
```

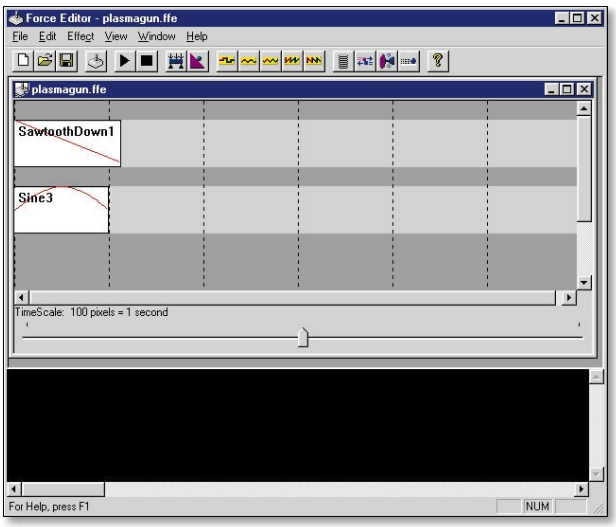
Damit haben Sie ein gewünschtes Eingabegerät angefordert. Nun müssen Sie den Geräten noch mitteilen, wie Sie von ihnen Daten erhalten wollen. Dabei können Sie genau einstellen, wie und welche Daten Sie bei der Bewegung der Geräte bekommen. Sie teilen dieses DirectInput über die Syntax folgender Methode mit:

```
IDirectInputDevice:
➥:SetDataFormat
```

Dabei lönnen Sie das Datenformat komplett selbst definieren, indem Sie eine *DIDATAFORMAT*-Struktur füllen und als Parameter übergeben, oder Sie verwenden einfach eine der global vordefinierten Variablen:

```
c_dfDIKeyboard
c_dfDIMouse
c_dfDIMouse2
c_dfDIJoystick
c_dfDIJoystick2
```

Für unsere Zwecke genügen die vordefinierten Einstellungen, und so setzen Sie sie für unsere angeforderten Geräte ein:



DEN FORCE-FEEDBACK-EDITOR finden Sie im DirectX Developer Kit.

```
hr = _keyboard->SetDataFormat
( &c_dfDIKeyboard );
  if( FAILED(hr) )
    return false;

  hr = _mouse->SetDataFormat
( &c_dfDIMouse2 );
  if( FAILED(hr) )
    return false;

  hr = _joy->SetDataFormat
( &c_dfDIJoystick2 )
  if( FAILED(hr) )
    return false;
```

Nachdem Sie das Programmgerüst so weit aufgebaut haben, müssen Sie DirectInput mitteilen, in welchem Modus Sie die Geräte betreiben wollen. Dabei steht Ihnen der *DISCL_EXCLUSIVE*-Modus zur Auswahl, bei dem Sie anderen Applikationen keinen Zugriff mehr auf das von Ihnen in Beschlag genommene Eingabegerät zugestehen können, oder der *DISCL_NONEXCLUSIVE*-Modus. Wie der Namen verrät, ist der Zugriff anderer Anwendungen in diesem Modus gestattet. Die Einstellungen nehmen Sie mit folgender Methode vor:

```
IDirectInputDevice7:
➥:SetCooperativeLevel
```

Damit können Sie genau beeinflussen, wie Ihre Instanz des Eingabeobjekts mit eventuell vorhandenen anderen Instanzen und dem System interagiert. Sie können fordern, dass Ihre Anwendung im Vordergrund laufen muss, oder Sie deaktivieren die Windows-Taste. Der Aufruf sieht folgendermaßen aus:

```
hr = _keyboard->
SetCooperativeLevel( _hwnd,
DISCL_FOREGROUND |
DISCL_NONEXCLUSIVE );
```

Nachdem Sie das *Coorperative Level* gesetzt haben, müssen Sie das Eingabegerät vorbereiten und ihm mitteilen, dass Sie ab jetzt Daten bekommen wollen. Dies erledigen Sie mit der Syntax:

```
IDirectInputDevi-
ce7:
➥:Acquire Methode:

hr= _keyboard.devi-
ce->Acquire();
hr= _mouse.device-
>Acquire();
hr= _joy.device-
>Acquire();
```

Damit sind die Vorarbeiten, das Eingabegerät anzusteuern, abgeschlossen, und Sie können die Daten abholen. Diese liegen im gerätespezifischen Format vor, das Sie entweder aus der vordefinierten Liste gewählt oder selbst festgelegt haben. Dazu geben Sie beispielhaft ein:

```
_mouse->SetDataFormat
( &c_dfDIMouse2 ); )
```

In unserem Beispielprogramm holen Sie mit einem Timer die jeweils aktualisierten Daten der einzelnen Eingabegeräte in Intervallen von zehn Millisekunden ab. In einer Direct3D-Anwendung oder einem Spiel könnten Sie die Daten zum Beispiel auch nach jedem Frame, also nach jedem berechneten Bild, abholen. Dabei sollten Sie berücksichtigen, dass heutige 3D-Beschleuniger eventuell sehr

hohe Frame-Raten schaffen. Mit weit über 100 Bildern pro Sekunde werden die Zeitintervalle unnötig klein. Bei zu kleinen Zeitintervallen erhalten Sie zu viele redundante Daten, da DirectInput Ihnen den jeweiligen Ist-Zustand eines Eingabesystems liefert.

Einige Eingabesysteme wie Joysticks sind *pollable devices*. Das heißt, dass Sie DirectInput informieren müssen, bevor Sie aktualisierte Daten von einem dieser Geräte abholen. Der Befehl dazu

```
_joy.device->Poll()
```

wird von nicht *pollable devices* automatisch ignoriert.

Die Methode

```
IDirectInputDevice:
➥:GetDeviceState()
```

holt die Daten ab. Übergeben Sie ihr einen Zeiger auf eine Struktur, die die Daten aufnehmen soll, und die Größe dieser Struktur. Wenn Sie selbst definierte Datenformate gewählt haben, müssen Sie auch definieren, wie diese Struktur aussieht. Für die vordefinierten Datenformate sehen Sie in der Tabelle unten den Namen der jeweils zugehörigen Struktur.

Mit folgenden Zeilen lesen Sie die aktuellen Informationen eines Joysticks aus:

```
HRESULT hr;
// Struktur für die Daten
DIJOYSTATE2 js2;

  // Joystick-Info auslesen
  hr = DIERR_INPUTLOST;
  while( DIERR_INPUTLOST == hr)
  {
//Poll Befehl ausführen, um neue
// Daten auslesen zu können
    hr = _joy.device->Poll();

// neue Daten auslesen
    hr = _joy.device->
GetDeviceState( sizeof
(DIJOYSTATE2), &js2 );

    if( hr == DIERR_INPUTLOST )
      if( FAILED( _joy.device->
Acquire() ) )
      {
 _log( „inputWRAPPER_DX7:
updateJOY :
cannot GetDeviceState“ );
        return false;
      }
    }
```

Als Beispiel greifen Sie die *DIJOYSTATE2*-Struktur heraus:

```
typedef struct DIJOYSTATE2 {
 LONG lX; LONG lY; LONG lZ;
 LONG lRx; LONG lRy; LONG lRz;
 LONG rglSlider[2];
 DWORD rgdwPOV[4];
 BYTE rgbButtons[128];
 LONG lVX; LONG lVY; LONG lVZ;
 LONG lVRx; LONG lVRy;LONG lVRz;
 LONG rglVSlider[2];
 LONG lAX; LONG lAY; LONG lAZ;
 LONG lARx; LONG lARy;LONG lARz;
 LONG rglASlider[2];
 LONG lFX; LONG lFY; LONG lFZ;
 LONG lFRx; LONG lFRy;LONG lFRz;
 LONG rglFSlider[2];
} DIJOYSTATE2, *LPDIJOYSTATE2;
```

Das Listing macht deutlich, dass die Zeiten des simplen Atari Joysticks mit einem roten Knopf vorbei sind.

Force Feedback mit DirectInput

Mit dem Force-Feedback-Effekt-Editor (Bestandteil des DirectX7 SDK) nutzen Sie die Fähigkeiten der Force-Feedback-Geräte. Mit diesem Editor können Sie eigene Effekte erzeugen und speichern. Sie können diese Dateien dann mit DirectInput laden und für Ihre Force-Feedback-Geräte nutzen.

Laden Sie eine dieser Dateien, und holen Sie sich daraus die Effekte für den Joystick:

```
FILE_EFFECT effects;
  effects.device = _joy.device;
  char filename[] = „test.ffe“;
// Effekte Enumerieren im File
//und im Callback erzeugen
  if( _joy.canREACT ) {
    if( FAILED( _joy.device->
EnumEffectsInFile( file,
EnumAndCreateEffectsCallback,
&effects, 0 ) ) )
      return false;
  }
```

Um die Werte zu verarbeiten, benötigen Sie wieder eine Callback-Prozedur. Unsere Variante überprüft einfach, ob der angeschlossene Joystick die Fähigkeit mitbringt, einen Effekt abzuspielen, und fügt ihn in diesem Fall in eine Liste ein.

```
BOOL CALLBACK
EnumAndCreateEffectsCallback
(LPCDIFILEEFFECT fileEFFECT,
LPVOID pvRef ) {

  HRESULT hr;
  LPDIRECTINPUTEFFECT effect
  = NULL;
 FILE_EFFECT* entry =
reinterpret_cast<FILE_EFFECT*>
 ( pvRef );

  // den File-Effekt anlegen
  if( FAILED( hr = entry->
 device->CreateEffect(
  fileEFFECT->GuidEffect,
fileEFFECT->lpDiEffect,
&effect, NULL ) ) )

//effekt nicht wiederzugeben
//nächster bitte
    return DIENUM_CONTINUE;
// eine neue Effekt-Node anlegen
  EFFECT_NODE enode;
  enode.effect = effect;
  enode.repeats = 1;
//in verkettete Liste einhängen
  if( effect )
entry->effects.push_back
 ( enode );

  return DIENUM_CONTINUE;
}
```

Jetzt können Sie die geladenen Effekte, die Sie in der verketteten Liste gespeichert haben, ausgeben. Stoppen Sie alle eventuell schon laufenden Force-Feedback-Effekte:

```
//Zeiger auf Effekt d.Liste:
  FILE_EFFECT *feff = ...;
  feff->device->
SendForceFeedbackCommand
 (DISFFC_STOPALL) ;
```

Und so spielen Sie alle Effekte aus der Liste der Reihe nach ab:

```
//eleganter Iterator:
  std::vector<EFFECT_NODE>:
:iterator it;
for( it = feff->effects.begin();
it < feff->effects.end(); it++ )
    // und Effekt starten !
(*it).effect->Start( (*it)
 .repeats, 0 );
```

Mit diesem Instrumentarium können Sie Ihre Eingabegeräte umfassend nutzen. Damit spielen Sie auf allen Programmregistern dieser Eingabegeräte und – dank Force Feedback – auch Ausgabegeräte. DirectInput ist eine der Säulen des DirectX-Systems und damit wesentlicher Bestandteil jedes Spiels, das auf DirectX basiert. ET

Die Quelltexte sowie die fertig übersetzten Routinen finden Sie auf unserer Website unter www.pc-magazin.de/magazin/extras.htm.

Klicken Sie unter *Online Extras* im Menü *Praxis* auf das entsprechende *Download*-Feld.

DIRECTINPUT MIT SEINEN DEVICE-TYPES

```
DIDEVTYPE_MOUSE
    DIDEVTYPEMOUSE_UNKNOWN
    DIDEVTYPEMOUSE_TRADITIONAL
    DIDEVTYPEMOUSE_FINGERSTICK
    DIDEVTYPEMOUSE_TOUCHPAD
    DIDEVTYPEMOUSE_TRACKBALL

  DIDEVTYPE_KEYBOARD
    DIDEVTYPEKEYBOARD_UNKNOWN
    DIDEVTYPEKEYBOARD_PCXT
    DIDEVTYPEKEYBOARD_OLIVETTI
    DIDEVTYPEKEYBOARD_PCAT
    DIDEVTYPEKEYBOARD_PCENH
    DIDEVTYPEKEYBOARD_NOKIA1050
    DIDEVTYPEKEYBOARD_NOKIA9140
    DIDEVTYPEKEYBOARD_NEC98
    DIDEVTYPEKEYBOARD_NEC98LAPTOP
    DIDEVTYPEKEYBOARD_NEC98106
    DIDEVTYPEKEYBOARD_JAPAN106
    DIDEVTYPEKEYBOARD_JAPANAX
    DIDEVTYPEKEYBOARD_J3100

  DIDEVTYPE_JOYSTICK
    DIDEVTYPEJOYSTICK_UNKNOWN
    DIDEVTYPEJOYSTICK_TRADITIONAL
    DIDEVTYPEJOYSTICK_FLIGHTSTICK
    DIDEVTYPEJOYSTICK_GAMEPAD
    DIDEVTYPEJOYSTICK_RUDDER
    DIDEVTYPEJOYSTICK_WHEEL
    DIDEVTYPEJOYSTICK_HEADTRACKER

  DIDEVTYPE_HID
```