게임프로그래밍

입력

박종승

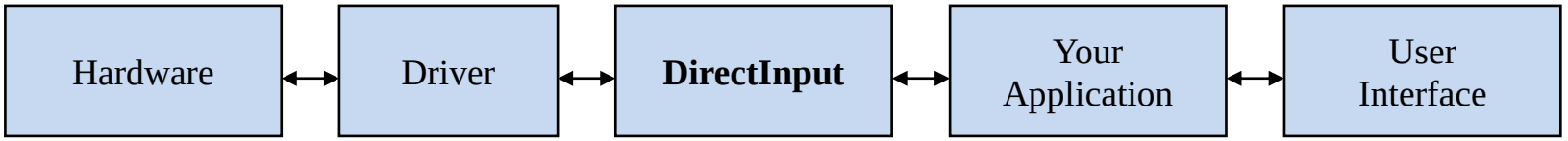
Dept. of CSE, Incheon Nat. Univ.
jong@inu.ac.kr
http://ecl.inu.ac.kr

목차

- DirectInput 개요
- DirectInput의 초기화
- 장치 객체 생성
- 장치의 설정
- 장치로부터 입력 데이터 얻기

Why DirectInput

- 윈도우 메시지는 게임에 충분하지 않음
  - 키를 일정시간 누르고 있는 경우
  - 입력장치의 상태를 직접 폴링(polling)할 수 없음
  - 지연시간이 큼
- DirectInput의 장점
  - 어떤 입력 장치라도, 어떤 특별한 capability라도, full access할 수 있음
  - 하드웨어에 대한 빠르고 직접적인 access가 가능함
  - DirectX의 추상화를 사용하므로 장치에 specific한 코드를 없앨 수 있음
- DirectInput 특징
  - 입력 장치의 직접 제어: 윈도우 드라이버와 직접 통신함
  - 운영체제의 윈도우 메시지 체계의 제한을 받지 않음

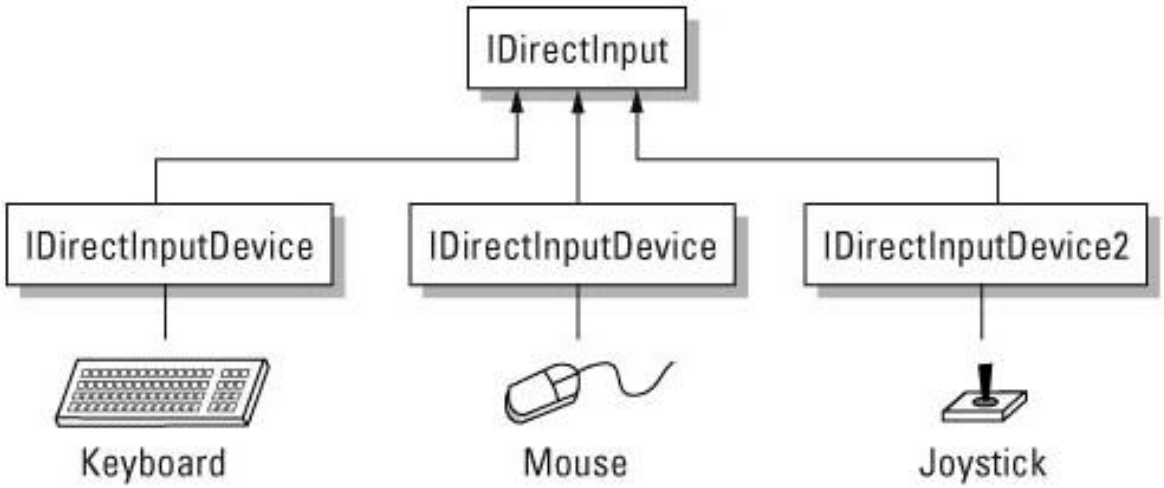


다양한 입력 장치

- 입력 장치들과 특별한 장치 capabilities
  - keyboard: 표준 QWERTY keyboard
  - mouse: 2-button 그리고 3-button mouse
  - Joystick과 비행 Yoke: 아날로그와 디지털 Joystick/비행-Yoke는 6만큼의 자유도와 최대 32개의 버튼을 지원한다.
  - 운전대 컨트롤: 아날로그 디지털 운전 시뮬레이션용
  - paddle: 다양한 회전 장치
  - force-feedback 장치: 기계적인 actuator가 있어서, 컴퓨터가 진동시키거나 모양을 바꿀 수 있는 joystick 또는 다른 장치들.
  - virtual reality headgear tracking system: 가상현실 headgear의 위치와 회전을 지원. (착용자의 머리의 상태를 전송.)

DirectInput 객체

- DirectInput 인터페이스의 구조
  - DirectInput 객체: 최상위 루트 DirectInput 인터페이스
  - DirectInputDevice 객체: 한 장치를 대표하는 코드임
  - Device 객체: 각 장치(keyboard, mouse, joystick)가 하나씩의 Device임
    - 각 입력 장치마다 하나씩의 DirectInput 장치 객체를 생성
- DirectInput project setup
  - 헤더파일 : "dinput.h"
  - 링크파일 : "dinput8.lib", "dxguid.lib"



DirectInput 사용을 위한 단계

- 장치 객체의 생성
- 장치의 설정
- 장치로부터 입력 데이터 얻기
- 장치 반납

장치 객체의 생성

- IDirectInput8 객체 얻기
  - DirectInput 객체 : IDirectInput8
  - DirectInput 객체의 한 reference 얻기
    - DirectInput8Create() 함수를 호출

```
LPDIRECTINPUT8 pDirectInput;
DirectInput8Create(
        appInstance, DIRECTINPUT_VERSION, IID_IDirectInput8,
        &pDirectInput, NULL );
```

  - 인자
    - appInstance: the handle to your application
      - 흔히 GetModuleHandle(NULL) 로 얻음
    - DIRECTINPUT_VERSION: 현재의 DirectInput 버전
    - IID_IDirectInput8: 생성할 인터페이스명이 DirectInput8 임을 의미함
    - pDirectInput : 리턴할 DirectInput 객체 포인터

장치 객체의 생성

- 가용한 장치들 알아보기 (Optional)
  - 함수 IDirectInput8::EnumDevices()
  - 예

```
pDirectInput->EnumDevices(
            DI8DEVCLASS_GAMECTRL,
            DeviceCallback, NULL,
            DIEDFL_ATTACHEDONLY | DIEDFL_FORCEFEEDBACK )
```

  - 인자
    - dwDevType: 관심있는 장치 타입을 명시. DI8DEVCLASS_* 중 하나를 명시.
      - DI8DEVCLASS_GAMECTRL: 모든 게임 controller들
    - lpCallback: 각 나열되는 장치마다 한번씩 호출될 callback 함수
    - pvRef: callback 함수가 호출될 때에 넘겨줄 인자값
    - dwFlags: enumeration 범위를 명시함
      - DIEDFL_ATTACHEDONLY | DIEDFL_FORCEFEEDBACK
        - » attached and force feedback device들만 나열하라는 의미

장치 객체의 생성

- IDirectInputDevice8 객체 얻기
  - 장치 객체 : IDirectInputDevice8
  - 각 장치에 대해서 하나씩의 장치 객체를 생성
  - 장치 객체의 생성 함수: DirectInput8::CreateDevice()
    - 각 호출에 대해서 GUID가 필요함
    - GUID_SysKeyboard, GUID_SysMouse, …

- 디폴트 시스템 키보드에 대한 장치 생성하기

```
LPDIRECTINPUTDEVICE8 pKeyboard;
pDirectInput->CreateDevice(GUID_SysKeyboard, &pKeyboard, NULL);
```

- 디폴트 마우스에 대한 장치 생성하기

```
LPDIRECTINPUTDEVICE8 pMouse;
pDirectInput->CreateDevice(GUID_SysMouse, &pMouse, NULL);
```

장치의 설정

- 장치의 capability 얻기 (Optional)
  - 함수 IDirectInputDevice8::GetCapabilities() 를 호출
  - 예
    - pDevice->GetCapabilities(&DICaps)
      - where DICaps is a DIDEVCAPS structure
  - 리턴하는 DIDEVCAPS 구조체 내부에 여러 정보들이 있음
    - 장치 타입, 축의 개수, 버튼의 개수, …

장치의 설정

- 협력레벨(Cooperative Level)의 설정 (highly recommended)
  - 목적: let DirectInput know how this device should interact with the system and with other DirectInput applications.
  - IDirectInputDevice8::SetCooperativeLevel(hwnd, dwFlags)
  - 인자 hwnd: 이 응용의 최상위 레벨 윈도우 핸들
  - 인자 dwFlags
    - DISCL_FOREGROUND (↔DISCL_BACKGROUND)
      - 이 응용이 foreground일 때에만 장치 데이터에 접근할 수 있음.
      - 명시된 윈도우가 background로 될 때에 장치가 자동적으로 unacquire됨.
    - DISCL_EXCLUSIVE (↔ DISCL_NONEXCLUSIVE)
      - 이 응용이 장치를 acquire한 후에는 다른 어떤 응용도 이 장치에 대한 exclusive 접근을 요청할 수 없음. (nonexclusive 접근 요청은 할 수 있음).
  - 예

```
pKeyboard->SetCooperativeLevel(
            hwnd, // handle to your top-level window
            DISCL_FOREGROUND | DISCL_NONEXCLUSIVE );
```

장치의 설정

- 데이터 포맷 지정 (required)
  - 기능
    - 데이터 포맷은 장치에서 우리가 관심있는 제어값들이 무엇인지,
    - 그 값들이 어떻게 준비되어야 하는지를 명시함.
  - IDirectInputDevice8::SetDataFormat(lpdf)
  - 인자: DIDATAFORMAT
    - 사용자가 명시한 구조체값을 사용하든지, 또는 미리 정의된 구조체값을 사용
  - 흔한 장치들에 대한 미리 정의된 DIDATAFORMAT 변수
    - c_dfDIKeyboard: An array of 256 characters
    - c_dfDIMouse: a mouse of three axes and four buttons
    - c_dfDIJoystick: a joystick
    - c_dfDIMouse2, c_dfDIJoystick2
  - 예

```
pKeyboard->setDataFormat(&c_dfDIKeyboard);
pMouse->setDataFormat(&c_dfDIMouse);
```

장치의 설정

- 장치의 속성 지정
  - 장치의 행위를 정의하는 속성값들을 명시함
    - 입력 버퍼 크기, joystick의 dead/saturation zone, 등
  - IDirectInputDevice8::SetProperty(rguidProp, pdiph)
  - 인자값: 입력 버퍼 크기 명시 (DIPROP_BUFFERSIZE)
    - IDirectInputDevice8::GetDeviceData()의 호출 중간에서,
      데이터가 소실되기 전까지 버퍼에 보관할 수 있는 데이터의 량.
    - 장치로부터 버퍼된 입력 데이터를 읽기 위해서 명시해야 함

장치의 설정

- 장치의 획득(acquire)
  - 장치로부터 데이터를 얻어오려는 시점 이전에 장치를 획득해야 함.
  - 일단 획득된 후에는 장치의 속성을 바꿀 수 없음.
  - IDirectInputDeivce8::Acquire()
    - 응용프로그램이 전면 상태로 될 때마다 매번 다시 장치를 획득해야 함
  - 예: pKeyboard->Acquire();
  - 장치를 foreground 접근하는 경우
    - 응용이 background로 될 때 (focus를 잃었을 때, 창이 최소화될 때), 자동적으로 장치가 unacquire됨.
    - 다시 foreground가 될 때에 그 장치를 다시 acquire하도록 해야 함.
      - 윈도우 메시지 WM_ACTIVATE 가 발생되는 시점에서 처리
- 장치의 해제(unacquire) (응용이 종료될 때에)
  - 응용이 종료되는 경우 또는 잠깐 장치로부터 입력 받지 않도록 하는 경우에 수행.
  - IDirectInputDevice::Unacquire()
  - 예: pKeyboard->Unaquire();
  - 이후에 Acquire()를 다시 수행할 수 있음.

장치로부터 입력 데이터 얻기

- 장치로부터 입력 데이터 얻기
  - IDirectInputDevice8::GetDeviceState()
    - 현재 상태가 미리 정의된 데이터 구조로 리턴됨
  - GetDeviceState() 함수의 인자 타입
    - 키보드: char[256]
    - 마우스: DIMOUSESTATE
  - 폴링이 필요한 장치(특정 joystick 등)인 경우,
    - GetDeviceState() 호출 전에 Poll()을 호출해야 입력 데이터가 버퍼에 준비됨.
  - 참고: Win32에서의 유사 기능 함수
    - 키보드 입력 얻기: GetAsyncKeyState()
    - 마우스 커서 위치 얻기: GetCursorPos(), ScreenToClient()

키보드 입력 얻기

- 형식

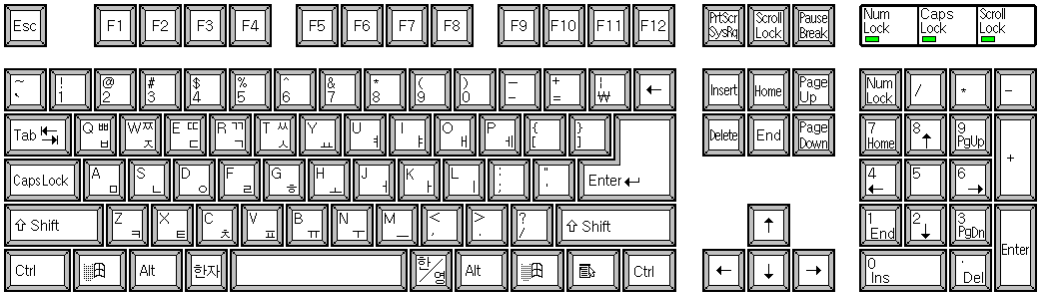
```
UCHAR keyState[256]; // storage for the keyboard data
pKeyboard->GetDeviceState(sizeof(keyState), (LPVOID)keyState);
```

- 예

```
int ship_x = 100, ship_y = 100; // initial ship position
UCHAR keyState[256];
// in your main loop, you would do this
if (pKeyboard->GetDeviceState(256, keyState)!=DI_OK) { /*error*/ }
// test whether the player is "moving" the ship
if (keystate[DIK_RIGHT ] & 0x80) ship_x++;
if (keystate[DIK_LEFT  ] & 0x80) ship_x--;
if (keystate[DIK_DOWN] & 0x80) ship_y++;
if (keystate[DIK_UP    ] & 0x80) ship_y--;
```

키보드 입력 얻기'

- DirectInput 키보드 가상 키 코드
  - DIK_ESCAPE ESC키
  - DIK_0 - DIK_9 메인 키보드의 0-9
  - DIK_A - DIK_Z A-Z
  - DIK_RETURN 메인 키보드의 Enter key
  - DIK_LCONTROL 왼쪽 Ctrl키
  - DIK_RCONTROL 오른쪽 Ctrl키
  - DIK_SPACE spacebar
  - DIK_F1 - DIK_F12 function key F1-F12
  - DIK_UP (위)화살표
  - DIK_DOWN (아래)화살표
  - DIK_LEFT (왼쪽)화살표
  - DIK_RIGHT (오른쪽)화살표
  - DIK_PRIOR Page Up
  - DIK_NEXT Page Down



마우스 입력 얻기

- 형식

```
typedef struct DIMOUSESTATE {
  LONG lX; // x-axis
  LONG lY; // y-axis
  LONG lZ; // z-axis (wheel in mouse cases)
  BYTE rgbButtons[4]; //buttons, high bit means down
} DIMOUSESTATE, *LPDIMOUSESTATE;

DIMOUSESTATE mouseState; //will hold the mouse data
pMouse->GetDeviceState(sizeof(DIMOUSESTATE), &mouseState);
```

- 예

```
DIMOUSESTATE mouseState;
if (pMouse->GetDeviceState(sizeof(DIMOUSESTATE),
            (LPVOID)&mouseState)!=DI_OK) { /* problem */ }
// test left button
if (mouseState.rgbButtons[0])  { /* do whatever */ }
else if (mouseState.rgbButtons[1])  { /* do whatever */ }
// Is the mouse moving right?
if (mouseState.lX > 0 )  { /* right */ }
else /* check for left motion */ if (mouse_state.lX < 0) { /* left */ }
// Is the mouse moving down?
if (mouse_state.lY > 0 )   { /* down */ }
else /* check for upward motion */ if (mouse_state.lY < 0) { /* up */ }
```

Buffered Data

- 입력 데이터를 얻는 두 가지 형태
  - Immediate data: 장치의 현재 입력 상태값
    - GetDeviceState() 함수는 immediate 데이터를 리턴함
    - 이 방법은 대부분의 응용에서 충분함
      - DirectInput에서 디폴트로 이 방법을 사용함.
    - 단점: 프레임율이 느린 컴퓨터에서는, 입력이 lost될 수 있음.
  - Buffered data: 입력 버퍼에 누적되어 저장된 입력 상태값들
    - 버퍼 크기 내에서, 모든 입력들을 놓치지 않고 처리할 수 있음.
    - 이 형태를 사용하기 위해서는, 반드시, SetProperty() 함수를 호출하여,
    - 버퍼 크기를 양수로 명시해야 함.
    - GetDeviceData() 함수를 사용
  - GetDeviceData() 함수 사용의 예

```
DIDEVICEOBJECTDATA didod[SAMPLE_BUFFER_SIZE]; //Receive buffered data
DWORD dwElements = SAMPLE_BUFFER_SIZE;
pKeyboard->GetDeviceData(  sizeof(DIDEVICEOBJECTDATA),  // 구조체의 바이트 수
                didod,  // 버퍼된 데이터를 받기 위한 구조체 타입의 배열
                &dwElements,  // 입력: 배열의 원소의 개수; 출력: 채워진 원소의 개수;
                0 ); // 0: 버퍼에서 데이터를 제거함; DIGDD_PEEK: 버퍼에 남겨둠;
```

응용 종료하기

- 모든 장치들을 해제
  - pKeyboard->Unacquire();
  - pMouse->Unacquire();
- 장치들을 포함한 모든 COM 객체들을 반납
  - pKeyboard->Release();
  - pMouse->Release();
  - pDirectInput->Release();

예제

- 키보드 간단한 예제
- 마우스 간단한 예제

01.KeyboardSimple

03.MouseSimple

- 키보드 큐브 예제
- 마우스 큐브 예제
- 키보드 및 마우스 스카이박스 예제

02.KeyboardCube

04.MouseCube

05.KeyboardMouseSkybox

*참고
-DirectInput: Keyboard/Mouse (WKTai/GAME Lab./CSIE/NDHU):
http://game.csie.ndhu.edu.tw/Teaching/ComputerGraphicsAndGame/03-DirectInput-KbMs.pdf
-Part of "Game Programming Gurus":
http://www.yaldex.com/games-programming/0672323699_ch09lev1sec2.html