게임프로그래밍

XInput

박종승

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

목차

- XInput 개요
- XInput 준비
- XInputGetState

XInput이란?

- XInput은 Xbox 360 controller를 다루기 위한 API임
  - Windows 응용 및 Xbox 360 응용 모두에서 사용함
- XInput이 다루는 장치의 범위
  - Xbox 360과 연결하여 사용할 수 있는 모든 controller를 대상으로 함
  - 장치의 예: game pads, guitars, big-button controllers, drums, …
- XInput은 다음의 기능도 지원함
  - controller vibrations (force-feedback)
  - voice input and output using the Xbox 360 headset
- 모든 Xbox 360 controller들은 Windows(XP이상)와 호환됨
  - USB 장치로 인식됨
  - 고전적 게임 controller로 사용될 수도 있음 (즉 DirectInput 장치로도 인식됨)

XInput vs. DirectInput

- 게임 controller에 대해서는 XInput을 써라
  - 게임 controller에 대해서 DirectInput을 대체할 API임
- XInput의 장점 (DirectInput에 상대적인)
  - 사용하기 쉬움
  - 설정하기 및 입력을 검사하기가 빠름
  - Windows PC 및 Xbox 360 console 모두에서 사용 가능함
  - Xbox controller의 vibration 기능을 설정할 수 있음
  - 앞으로 출시될 Xbox controller들과도 작동될 것임
- DirectInput을 사용해도 됨
  - DirectInput으로도 Xbox controller의 입력을 다룰 수 있음
  - 그러나 제한점이 있음
    - Vibration 기능을 접근하지 못함
    - Left/right trigger를 구분하지 못하고 한 동일 버튼으로 인식함
    - Xbox 360 headset으로부터의 audio를 접근하지 못함

XInput 준비

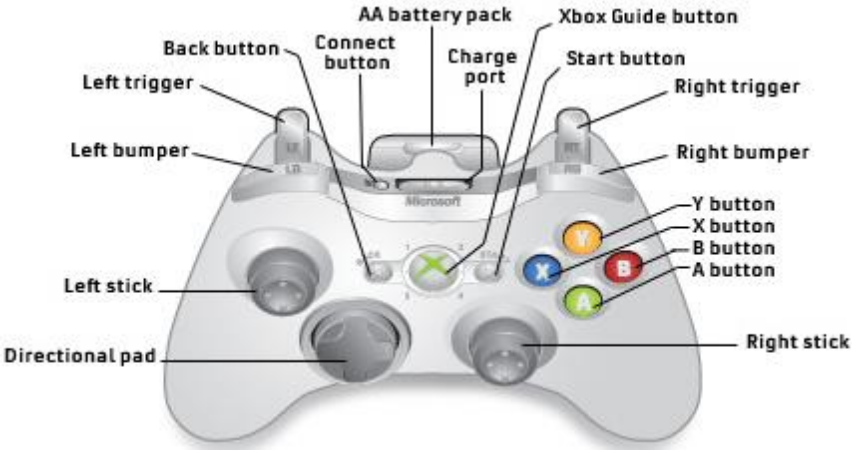
- 준비
  - 헤더 파일: XInput.h
  - 라이버러리 파일: xinput.lib

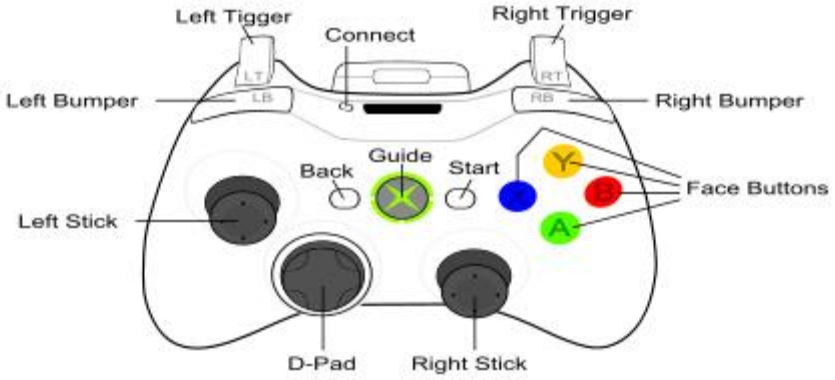
설정하기

- 설정을 위한 코드가 필요없음
  - Controller를 USB 장치로 인식되도록 PC에 연결만 하면 됨
  - 연결 후에 XInputGetState() 함수를 호출하면 됨
- XInput의 보고 상태를 설정할 수 있음
  - 보고 상태(reporting state)가 true이면: XInputGetState가 장치의 상태를 모두 리턴함
    - 디폴트: true
  - False이면: XInputGetState가 장치의 실제 상태와 무관한 neutral data만 리턴함
    - 응용이 focus를 잃은 경우(최소화된 경우)에 false로 명시하여 disable시키면 됨
    - neutral data: all buttons up, axes centered, and triggers at 0
  - 설정하는 방법: void XInputEnable(BOOL enable);

설정하기

- Xbox 360 game pad에 포함된 버튼들
  - four face buttons
  - two triggers
  - two bumper buttons (buttons in front of the triggers)
  - two analog sticks, each with a digital button (accessed by pushing the stick inward)
  - a guide button
  - a directional pad
  - start and back buttons.





XInputGetState

- XInput 입력 함수: XInputGetState()
  - Xbox controller의 상태를 얻는데 사용됨
  - 인자1: DWORD dwUserIndex
    - 플레이어의 인덱스(최대 4명을 위한 인덱스: 0,1,2,3)
  - 인자2: XINPUT_STATE* pState
    - 리턴할 상태값들을 위한 구조체
  - 리턴값: DWORD – controller가 연결되어 있는지의 여부를 알 수 있음
    - ERROR_SUCCESS: 해당 플레이어 인덱스를 위한 연결된 controller가 있음
    - ERROR_DEVICE_NOT_CONNECTED: 연결된 controller가 없음

```
DWORD dwResult;
for (DWORD i=0; i< MAX_CONTROLLERS; i++ ) {
        XINPUT_STATE state;
        ZeroMemory( &state, sizeof(XINPUT_STATE) );

        // Simply get the state of the controller from XInput.
        dwResult = XInputGetState( i, &state );

        if( dwResult == ERROR_SUCCESS ) { /* Controller is connected  */ }
        else { /* Controller is not connected */ }
}
```

XInputGetState

- XINPUT_STATE 구조체
  - 호출 시점에서의 장치의 상태값들을 저장하기 위한 구조체임
  - 필드들
    - DWORD dwPacketNumber
      - 상태 패킷의 번호임. Controller의 상태에 변경이 있으면 새로운 번호로 상태 패킷을 만들어서 리턴함. 따라서 패킷 번호가 이전 최근 호출과 동일하다면 상태에 변화가 없다는 것을 의미함.
    - XINPUT_GAMEPAD Gamepad
      - Xbox 360 Controller의 현재 상태를 가지는 구조체
- XINPUT_GAMEPAD의 필드들
  - WORD wButtons;
  - BYTE bLeftTrigger, bRightTrigger;
  - SHORT sThumbLX, sThumbLY, sThumbRX, sThumbRY;

버튼 상태 detect하기

- XINPUT_STATE 구조체 필드 – wButtons
  - 버튼과 관련된 모든 정보를 가지는 bitmask 값임
- Xbox controller 의 버튼 입력 상태를 표현하기 위한 flag들
  - XINPUT_GAMEPAD_XXX, XXX=
  - DPAD_UP, DPAD_DOWN, DPAD_LEFT, DPAD_RIGHT
  - START, BACK
  - LEFT_THUMB, RIGHT_THUMB, LEFT_SHOULDER, RIGHT_SHOULDER
  - A, B, X, Y
- 버튼 눌림을 detect하기
  - XInputGetState 함수로 상태를 얻은 후에 각 개별 버턴의 누름 상태를 테스트할 수 있음
  - 각 버튼의 상태는 XINPUT_STATE 구조체의 GamePad.wButtons 필드에 저장되어 있음
    - wButtons는 short int (또는 WORD)임
    - 각 비트가 각 버튼을 표현함

버튼 상태 detect하기'

```
XINPUT_STATE state;
XInputGetState(0, &state);

if(state.Gamepad.wButtons & XINPUT_GAMEPAD_LEFT_SHOULDER)    // left bumper
button was pressed…
if(state.Gamepad.wButtons & XINPUT_GAMEPAD_RIGHT_SHOULDER)    // right
bumper button pressed…

if(state.Gamepad.wButtons & XINPUT_GAMEPAD_BACK)    //back button pressed…
if(state.Gamepad.wButtons & XINPUT_GAMEPAD_START)    //start button pressed…
if(state.Gamepad.wButtons & XINPUT_GAMEPAD_A)    //face button 'a' pressed…
if(state.Gamepad.wButtons & XINPUT_GAMEPAD_B)    // face button 'b' pressed…
if(state.Gamepad.wButtons & XINPUT_GAMEPAD_X)    // face button 'x' pressed…
if(state.Gamepad.wButtons & XINPUT_GAMEPAD_Y)    // face button 'y' pressed…
```

버튼 상태 detect하기”

- Directional(arrow) pad
  - game pad controller 및 다른 controller(guitar등)에 붙어 있음
  - 버튼과 동일한 방식으로 다루면 됨
    - Up/down/left/right 화살표 버튼을 독립적으로 취급

```
if(state.Gamepad.wButtons & XINPUT_GAMEPAD_DPAD_UP)    // Do Something...
if(state.Gamepad.wButtons & XINPUT_GAMEPAD_DPAD_DOWN)    // Do Something...
if(state.Gamepad.wButtons & XINPUT_GAMEPAD_DPAD_LEFT)    // Do Something...
if(state.Gamepad.wButtons & XINPUT_GAMEPAD_DPAD_RIGHT)    // Do Something...
```

- left and right thumb stick’s digital buttons
  - Joystick을 thumbstick이라고도 함
  - Left/right joystick의 중간을 누르는 것이 thumb button을 누르는 것임

```
if(state.Gamepad.wButtons & XINPUT_GAMEPAD_LEFT_THUMB)// Do Something...
if(state.Gamepad.wButtons & XINPUT_GAMEPAD_RIGHT_THUMB)// Do Something...
```

Trigger 상태 detect하기

- Trigger란
  - Trigger는 일반 버튼과는 다름
  - Trigger는 압력에 반응하는 버튼임
  - XInput은 trigger에 가해지는 압력을 detect할 수 있도록 함
  - 현재 Xbox 360 controller 에는 2종류의 trigger가 있음:
    - Left trigger button, right trigger button
    - Game pad 장치 상단에 있음
- Trigger가 눌려졌는지를 test하는 방법
  - XINPUT_STATE 구조체의 Gamepad.bLeftTrigger 필드 / Gamepad.bRighttTrigger 필드
  - Type: unsigned char
  - Value: [0,255]; 0=not pressed, 255=pressed fully down

```
unsigned char lt = state.Gamepad.bLeftTrigger;
unsigned char rt = state.Gamepad.bRightTrigger;
```

Thumbstick 상태 detect하기

- Xbox 360 game pad
  - 두개의 thumb stick이 있음: left, right
- Thumb stick의 상태 표현
  - Gamepad.XXX; sThumbLX, sThumbLY, sThumbRX, sThumbRY
    - Left/right joystick의 x/y position의 표현을 위함
  - 값은 short int로 표현됨; 값의 범위: [-32,768, 32,767];
  - 값이 0이면 축의 중간에 있음 (joystick의 움직임이 없음을 의미함)
  - 예: LX값(왼쪽 joystick의 x축 값)의 경우; -32,768=stick이 완전히 왼쪽으로만 치우쳐 있음; 32,767=stick이 완전히 오른쪽으로만 치우쳐 있음;
  - LY값(왼쪽 joystick의 y축 값)의 경우; -327,68=stick이 완전히 아래쪽으로만 치우쳐 있음; 32,767=완전히 위쪽으로만 치우쳐 있음;

```
short lx = state.Gamepad.sThumbLX;
short ly = state.Gamepad.sThumbLY;
short rx = state.Gamepad.sThumbRX;
short ry = state.Gamepad.sThumbRY;
```

Dead Zone

- thumbstick의 상태
  - Gamepad.XXX; sThumbLX, sThumbLY, sThumbRX, sThumbRY
  - Joystick이 민감하여, 의도적인 움직임이 없음에도 불구하고 값이 정확이 0이 아닐 수 있음
  - 0 근처의 적정 구간(예: [-20,20])을 0으로 간주하자!
    - 이 구간을 dead zone이라고 함.
  - 상수가 정의되어 있음 (선택적으로 사용)
    - #define XINPUT_GAMEPAD_LEFT_THUMB_DEADZONE 7849
    - #define XINPUT_GAMEPAD_RIGHT_THUMB_DEADZONE 8689
    - #define XINPUT_GAMEPAD_TRIGGER_THRESHOLD 30
- trigger의 상태
  - Dead zone과 유사하게 일정 값 이상의 경우에만 눌러진 것으로 간주하도록 하자!

Controller Vibrations

- Vibration 기능: 장치에 따라서 아무 동작도 하지 않을 수 있음
  - Xbox controller는2개의 vibration 모드를 가짐
  - 어떤 controller(예: guitar)는 vibration 지원하지 않음
- game pad controller
  - 2개의 모터를 가짐 (left motor, right motor)
  - Left motor: 저주파 진동을 위함 (예: 발자국)
  - Right motor: 고주파 진동을 위함 (예: 폭발)
- 진동 지정하기
  - 단계1: XINPUT_VIBRATION 구조체를 생성
    - XINPUT_VIBRATION 구조체의 필드:
      - WORD wLeftMotorSpeed;
      - WORD wRightMotorSpeed;
  - 단계2: Left/right motor speed를 [0, 65,535] 내의 값으로 명시
    - 0= 진동 안함; 65,535=100% 파워로 진동함;
  - 단계3: XInputSetState() 함수를 호출
    - DWORD XInputSetState(DWORD dwUserIndex, XINPUT_VIBRATION* pVibration);
- 진동 해제하기
  - Motor speed를 0으로 하여서 XInputSetState() 함수를 다시 호출

Controller Capabilities

- Controller의 capabilities를 얻기
  - XInputGetCapabilities() 함수 호출
  - Prototype: DWORD XInputGetCapabilities()
    - 인자1: DWORD dwUserIndex; //플레이어 인덱스
    - 인자2: DWORD dwFlags; //현재까지의 버전은 XINPUT_FLAG_GAMEPAD 만 허용함
    - 인자3: XINPUT_CAPABILITIES* pCapabilities; //출력 -- controller의 capabilities
- XINPUT_CAPABILITIES 구조체 필드
  - BYTE Type;
    - 항상 XINPUT_DEVTYPE_GAMEPAD 이어야 함
  - BYTE SubType;
    - XINPUT_DEVSUBTYPE_ARCADE_STICK : arcade stick controller
    - XINPUT_DEVSUBTYPE_GAMEPAD : game pad controller
      - Guitar 등의 그 밖의 controller들
    - XINPUT_DEVSUBTYPE_WHEEL : steering wheel
  - WORD Flags;
    - XINPUT_CAPS_VOICE_SUPPORTED 만 허용함
  - XINPUT_GAMEPAD Gamepad;
    - Controller 버튼, thumb stick, 그 외의 상태들을 저장함
  - XINPUT_VIBRATION Vibration;
    - 장치의 현재 진동 상태를 저장함

Headset

- Xbox 360 controller에 연결된 headset의 제어
  - 두 기능: Microphone을 사용한 사운드 레코딩, Headphone으로 사운드 재생
- XInput 1.3 (Windows 7)
  - 이들 기능은 DirectSound로 구현됨: IDirectSound8, IDirectSoundCapture8

```
XInputGetDSoundAudioDeviceGuids( i, &dsRenderGuid, &dsCaptureGuid );

// Create IDirectSound8 using the controller's render device
if( FAILED( hr = DirectSoundCreate8( &dsRenderGuid, &pDS, NULL ) ) )    return hr;
// Set coop level to DSSCL_PRIORITY
if( FAILED( hr = pDS->SetCooperativeLevel( hWnd, DSSCL_PRIORITY) ) )    return hr;
// Create IDirectSoundCapture using the controller's capture device
if( FAILED( hr = DirectSoundCaptureCreate8( &dsCaptureGuid, &pDSCapture, NULL ) ) )   return hr;
```

- XInput 1.4 (Windows 8)
  - 이들 기능은 Windows Audio Session API (WASAPI)으로 구현됨

```
WCHAR renderID[256] = {0};
WCHAR captureID[256] = {0};
UINT renderCount = 256;
UINT captureCount = 256;
XInputGetAudioDeviceIds( 0, renderID, &renderCount, captureID, &captureCount );

IXAudio2* pXAudio2 = nullptr;
if ( *renderID == 0 || FAILED(hr = XAudio2Create( &pXAudio2, 0 ) ) ) return hr

IXAudio2MasteringVoice* pMasteringVoice = nullptr;
if ( FAILED( hr = pXAudio2->CreateMasteringVoice( &pMasteringVoice, 0, 0, 0, renderID ) ) ) return hr;
```

예제

- 간단한 XInput 예제

04.XInputSimple

참고: 추가 학습

- Windows에서의 사용자 상호작용 기법들 (link)
- XInput (link)