

DirectInput

En esta clase que vamos a desarrollar para manejar el componente de DirectX que permite manejar los dispositivos de E/S llamado DirectInput sólo vamos a implementar funcionalidades para controlar el teclado y el ratón. Quizá en un futuro se incluya la posibilidad de controlar joysticks y *force feedback*.

Hecha la introducción ya podemos empezar. Los pasos básicos a seguir para utilizar DInput son los siguientes:

- Lo primero de todo es crear el objeto DInput con el cual crearemos los dispositivos necesarios (teclado, ratón, etc).
- Hecho esto ya podemos crear el dispositivo que queramos, es decir, el teclado, el ratón, etc.
- Una vez que tenemos el dispositivo creado especificamos el formato de datos del dispositivo en cuestión y especificamos el nivel cooperativo.
- El siguiente paso es adquirir el dispositivo.
- Finalmente, en nuestra función de actualización debemos comprobar que no hemos perdido el control sobre el dispositivo y si ha ocurrido eso entonces lo intentamos recuperar. También es aquí donde se actualiza el estado del dispositivo, es decir, en el caso del teclado ver qué tecla/s ha sido pulsada, en el caso del ratón las coordenadas del mismo y si se ha pulsado alguno de sus botones, etc.

Es importante recordar que debemos incluir la librería **dinput8.lib** en nuestras opciones de link y el archivo de cabecera **dinput.h** de esta manera en el código fuente para que no nos avise de un error al compilar:

```
#define DIRECTINPUT_VERSION 0x0800
#include <dinput.h>
```

Veamos en primer lugar la definición de nuestra clase:

```
enum MOUSE_BUTTON
{
    BOTON_IZQUIERDO,
    BOTON_DERECHO,
    BOTON_CENTRAL
};

//-----
// Clase: CInput
//-----

class CInput{

private:

    HRESULT EstadoDelTeclado(void);
    HRESULT EstadoDelRaton(void);

    LPDIRECTINPUT8 ObjDinput;
    LPDIRECTINPUTDEVICE8 Teclado;
    LPDIRECTINPUTDEVICE8 Raton;

    BYTE Teclas[256];
    DIMOUSESTATE2 EstadoRaton;
```

```

int EstIzqdo,EstDcho,EstCentral;

public:

    BOOL PULSADO_IZQUIERDO;
    BOOL PULSADO_DERECHO;
    BOOL PULSADO_CENTRAL;

    BOOL SOLTADO_IZQUIERDO;
    BOOL SOLTADO_DERECHO;
    BOOL SOLTADO_CENTRAL;

    CInput();
    ~CInput();

    HRESULT Inicializar(HWND hWnd,BOOL RatonExclusivo,BOOL TecladoExclusivo);
    HRESULT Actualizar(void);

    BOOL TeclaPulsada(int iKey);

    BOOL BotonPulsado(int iButton);

    int XRelativaRaton(void);
    int YRelativaRaton(void);
    int ZRelativaRaton(void);

    int XAbsolutaRaton(void);
    int YAbsolutaRaton(void);
    int ZAbsolutaRaton(void);

};

```

Primero se han definido unas constantes que representan los 3 botones estándar de un ratón mediante un enumerado: el botón izquierdo, el derecho y el central. Hay ratones que tienen más pero lo normal son tres o incluso dos. Es importante que seguir ese orden ya que el dispositivo *ratón* de DInput contiene un array con el estado de los botones y en ese array, los botones van en ese orden: **BOTON_IZQUIERDO**, **BOTON_DERECHO**, **BOTON_CENTRAL**, etc.

Observemos que en la zona privada de la clase se han declarado dos funciones que lo que hacen es actualizar el estado de los dispositivos de teclado y ratón: **EstadoDelTeclado()** y **EstadoDelRaton()**. En ellas también se controla que no se haya perdido el dispositivo y si así ha sido se intenta recuperarlo.

También nos encontramos con el objeto DInput (**ObjDinput**), los dispositivos de teclado (**Teclado**) y de ratón (**Raton**), un array que contiene las 256 teclas de un teclado estándar (**Teclas**) y por último una estructura de estado del ratón (**EstadoRaton**).

En la sección pública, podemos observar que hay varias funciones. Las más interesantes son **Inicializar()** y **Actualizar()**. **Inicializar()** lo que hace es crear el objeto DInput y los dispositivos de teclado y de ratón y **Actualizar()** lo único que hace es llamar a las funciones privadas **EstadoDelTeclado()** y **EstadoDelRaton()** y actualizar las variables que nos dicen la pulsación única de un botón del ratón, pero esto se explicará más adelante. Las demás funciones de la clase son bastante obvias y son fáciles de entender. También encontramos varias variables booleanas que nos indicarán si un botón del ratón ha sido pulsado únicamente (es decir, hasta que no se suelte el botón no dará señal de ser pulsado otra vez).

Ahora pasaremos a ver la implementación de algunas de las funciones de la clase. Comenzaremos con **Inicializar()**:

```
//-----  
// Función: Inicializar  
// Propósito: Inicializa DInput, realiza la creación del teclado y del ratón.  
//-----  
  
HRESULT CInput::Inicializar(HWND hWnd, BOOL RatonExclusivo, BOOL TecladoExclusivo)  
{  
    ///////////////////////////////////////  
    // Creamos el objeto DInput  
    if(FAILED(DirectInput8Create(GetModuleHandle(NULL), DIRECTINPUT_VERSION,  
IID_IDirectInput8, (VOID**)&ObjDinput, NULL)))  
    {  
        MessageBox(hWnd, "¡No se pudo inicializar DInput!", "Error",  
MB_OK|MB_ICONEXCLAMATION);  
        return E_FAIL;  
    }  
    ///////////////////////////////////////  
  
    ///////////////////////////////////////  
    // Creamos el device Keyboard  
    if(FAILED(ObjDinput->CreateDevice(GUID_SysKeyboard, &Teclado, NULL)))  
    {  
        MessageBox(hWnd, "¡No se pudo crear el objeto teclado!", "Error",  
MB_OK|MB_ICONEXCLAMATION);  
        return E_FAIL;  
    }  
  
    // Especificamos el formato de los datos  
    if(FAILED(Teclado->SetDataFormat(&c_dfDIKeyboard)))  
    {  
        MessageBox(hWnd, "¡No se pudo especificar el formato de datos del  
teclado!", "Error", MB_OK|MB_ICONEXCLAMATION);  
        return E_FAIL;  
    }  
  
    // Especificamos el modo cooperativo  
    if(TecladoExclusivo)  
    {  
        if(FAILED(Teclado->SetCooperativeLevel(hWnd,  
DISCL_EXCLUSIVE|DISCL_FOREGROUND)))  
        {  
            MessageBox(hWnd, "¡No se pudo especificar el formato de datos del  
teclado!", "Error", MB_OK|MB_ICONEXCLAMATION);  
            return E_FAIL;  
        }  
    }  
    else  
    {  
        if(FAILED(Teclado->SetCooperativeLevel(hWnd,  
DISCL_NONEXCLUSIVE|DISCL_FOREGROUND)))  
        {  
            MessageBox(hWnd, "¡No se pudo especificar el formato de datos del  
teclado!", "Error", MB_OK|MB_ICONEXCLAMATION);  
            return E_FAIL;  
        }  
    }  
}
```

```

}

// Adquirimos device keyboard
Teclado->Acquire();
////////////////////////////////////

////////////////////////////////////
// Creamos el device Mouse
if(FAILED(ObjDinput->CreateDevice(GUID_SysMouse, &Raton, NULL)))
{
    MessageBox(hWnd, "¡No se pudo crear el objeto mouse!", "Error",
MB_OK|MB_ICONEXCLAMATION);
    return E_FAIL;
}

// Especificamos el formato de los datos
if(FAILED(Raton->SetDataFormat(&c_dfDIMouse2)))
{
    MessageBox(hWnd, "¡No se pudo especificar el formato de datos del mouse!",
"Error", MB_OK|MB_ICONEXCLAMATION);
    return E_FAIL;
}

// Especificamos el modo cooperativo
if (RatonExclusivo)
{
    if(FAILED(Raton->SetCooperativeLevel(hWnd, DISCL_EXCLUSIVE
|DISCL_FOREGROUND)))
    {
        MessageBox(hWnd, "¡No se pudo especificar el formato de datos del
mouse!", "Error", MB_OK|MB_ICONEXCLAMATION);
        return E_FAIL;
    }
}
else
{
    if(FAILED(Raton->SetCooperativeLevel(hWnd, DISCL_NONEXCLUSIVE
|DISCL_FOREGROUND)))
    {
        MessageBox(hWnd, "¡No se pudo especificar el formato de datos del
mouse!", "Error", MB_OK|MB_ICONEXCLAMATION);
        return E_FAIL;
    }
}

// Adquirimos device mouse
Raton->Acquire();
////////////////////////////////////

return S_OK;
}

```

Como se puede ver, primero se crea el objeto DInput y una vez hecho esto se crea el dispositivo del teclado, se especifica el formato de datos, se especifica el nivel cooperativo y finalmente se adquiere el dispositivo. Lo mismo hacemos pero con el ratón. Evidentemente no nos vamos a poner a explicar las funciones de DInput para hacer todo esto ya que vienen perfectamente detalladas en la ayuda del SDK. De todas maneras no debe ser difícil de entender después de las cosas que hemos dado con los tutoriales

de Direct3D :)

Ahora vamos a ver la función que actualizaba los datos, es decir, **Actualizar()**:

```
//-----  
// Función: Actualizar  
// Propósito: Actualiza el teclado y el ratón en cada FRAME.  
//-----  
HRESULT CInput::Actualizar(void)  
{  
    if(FAILED(EstadoDelTeclado()))  
    {  
        return E_FAIL;  
    }  
  
    if(FAILED(EstadoDelRaton()))  
    {  
        return E_FAIL;  
    }  
  
    ///////////////////////////////////  
  
    if (EstIzqdo==0)  
    {  
        if (BotonPulsado(BOTON_IZQUIERDO))  
        {  
            PULSADO_IZQUIERDO=FALSE;  
            SOLTADO_IZQUIERDO=FALSE;  
            EstIzqdo=1;  
        }  
        else  
        {  
            PULSADO_IZQUIERDO=FALSE;  
            SOLTADO_IZQUIERDO=TRUE;  
        }  
    }  
    else  
    {  
        if (!BotonPulsado(BOTON_IZQUIERDO))  
        {  
            EstIzqdo=0;  
            PULSADO_IZQUIERDO=TRUE;  
            SOLTADO_IZQUIERDO=TRUE;  
        }  
    }  
  
    ///////////////////////////////////  
  
    if (EstDcho==0)  
    {  
        if (BotonPulsado(BOTON_DERECHO))  
        {  
            PULSADO_DERECHO=FALSE;  
            SOLTADO_DERECHO=FALSE;  
            EstDcho=1;  
        }  
        else  
        {  
            PULSADO_DERECHO=FALSE;  
            SOLTADO_DERECHO=TRUE;  
        }  
    }  
}
```

```

    }
}
else
{
    if (!BotonPulsado(BOTON_DERECHO))
    {
        EstDcho=0;
        PULSADO_DERECHO=TRUE;
        SOLTADO_DERECHO=TRUE;
    }
}

////////////////////////////////////

if (EstCentral==0)
{
    if (BotonPulsado(BOTON_CENTRAL))
    {
        PULSADO_CENTRAL=FALSE;
        SOLTADO_CENTRAL=FALSE;
        EstCentral=1;
    }
    else
    {
        PULSADO_CENTRAL=FALSE;
        SOLTADO_CENTRAL=TRUE;
    }
}
else
{
    if (!BotonPulsado(BOTON_CENTRAL))
    {
        EstCentral=0;
        PULSADO_CENTRAL=TRUE;
        SOLTADO_CENTRAL=TRUE;
    }
}

////////////////////////////////////

return S_OK;
}

```

Lo primero que hacemos es llamar a **EstadoDelTeclado()** y **EstadoDelRaton()** para actualizar ambos dispositivos y acto seguido actualizamos los valores de las variables booleanas que nos dirán si un botón del ratón ha sido pulsado, pero lo dirán una única vez. Es decir, si pulsamos el botón izquierdo y en nuestro programa miramos el estado de **PULSADO_IZQUIERDO** la primera vez que se llame a la función **Actualizar()** esta variable tomará el valor **TRUE**, pero las siguientes serán **FALSE** mientras esté el botón pulsado (mientras esta el botón pulsado la variable **SOLTADO_IZQUIERDO** permanecerá a **FALSE**). Esto es útil para comprobar pulsaciones en menús, en donde queremos que sólo se realice una pulsación para ejecutar la orden.

Seguidamente vamos a ver el código de las funciones que hemos llamado desde **Actualizar()**. Empecemos con **EstadoDelTeclado()**:

```

//-----
// Función: EstadoDelTeclado
// Propósito: Actualiza el teclado rellenando el array de teclas.
//-----

```

```

HRESULT CInput::EstadoDelTeclado(void)
{
    HRESULT hr;

    if(!Teclado)
        return S_OK;

    // Recogemos el estado del keyboard y rellenamos el buffer de teclas
    ZeroMemory(&Teclas, sizeof(Teclas));
    if(FAILED(Teclado->GetDeviceState(sizeof(Teclas), &Teclas)))
    {
        // Si hemos perdido el device entonces intentamos recuperarlo
        hr = Teclado->Acquire();
        while(hr == DIERR_INPUTLOST)
            hr = Teclado->Acquire();

        return S_OK;
    }

    return S_OK;
}

```

Lo primero que hacemos es ver si tenemos un dispositivo teclado adecuado. Si lo tenemos continuamos. Entonces lo que hacemos a continuación es rellenar el array de teclas que nos dirá que tecla/s ha sido pulsada. También se averigua si hemos perdido el dispositivo. Si es así se intenta recuperar.

Ahora veamos como hacemos esto pero con el raton en la función **EstadoDelRaton()** que no es muy distinta a la anterior:

```

//-----
// Función: EstadoDelRaton
// Propósito: Actualiza el ratón rellenando la estructura de estado.
//-----
HRESULT CInput::EstadoDelRaton(void)
{
    HRESULT hr;

    if(!Raton)
        return S_OK;

    // Recogemos el estado del ratón y rellenamos la estructura de estado
    if(FAILED(Raton->GetDeviceState(sizeof(DIMOUSESTATE2), &EstadoRaton)))
    {
        // Si hemos perdido el device entonces intentamos recuperarlo
        hr = Raton->Acquire();
        while(hr == DIERR_INPUTLOST)
            hr = Raton->Acquire();

        return S_OK;
    }

    return S_OK;
}

```

La diferencia es que el estado del ratón no se almacena en un array de **BYTEs** sino que se almacena en una estructura especial (**DIMOUSESTATE2**).

Bueno, pues eso es lo más complejo que tiene la clase. Evidentemente es aconsejable mirar el resto de funciones.

Para concluir con el tutorial, pasaremos a mirar las cosas que hay que poner en nuestra aplicación para que CInput funcione correctamente. Para utilizar la clase lo primero es incluir si archivo de cabecera (CInput.h) y acto seguido crear el objeto de la clase (ya sea con punteros o no) en la parte del código que la vayamos a usar.

```
CInput g_Input;    // Objeto para manejo del teclado y ratón
```

Acto seguido habrá que inicializar el objeto pasándole el **hWnd** de nuestra ventana y los booleanos para los modos cooperativos del ratón y teclado (**TRUE** significa *modo exclusivo*). Esta llamada la podremos incluir por ejemplo en alguna función de inicialización del programa que tengamos:

```
g_Input.Inicializar(hWnd, FALSE, TRUE);
```

Por último hay que actualizar los datos de entrada en cada FRAME por lo tanto, al principio de nuestro bucle de actualización de pantalla tendrá que haber una llamada como la siguiente:

```
g_Input.Actualizar();
```

Pues ya tenemos lista para usar nuestra clase CInput.