

Tutorial 1: Init DirectInput

by Victor Saar

Content

In this tutorial we start programming a new class for all input devices. In the first tutorial we just init DirectInput. It is nearly the same procedure like with Direct3D. At the beginning we have to create the main DirectInput object. In the next tutorials we add the keyboard, the mouse and the joystick support to the class. We take the source code of the Direct3D tutorial two, so that we have a whole application. Later we take tutorial four, so that we can use the input to manipulate the scene.

For the new class we need two new files. One header and one source code file.

Before we can use DirectInput we have to link the project to *dinput8.lib* and *dxguid.lib*. As DirectInput wasn't updated with DirectX 9.0, we have to use DirectInput 8.0. The second library is needed for the creation of the main object.

cinput.h

We start with the definition of the input class. We call it `CInput` and declare three public methods. The constructor, the destructor and the function that inits DirectInput. Then we have the private DirectInput main object. These are the only things we need for the initialisation.

```
#ifndef cinput_h
#define cinput_h

#include "main.h"

class CInput
{
public:
    CInput(void);           //constructor
    ~CInput(void);         //destructor
    bool InitDirectInput(void); //inits DirectInput

private:
    LPDIRECTINPUT8 m_pDIObject; //DirectInput main object
};

#endif
```

First we have the constructor. We set the pointer to `NULL` and call `InitDirectInput()` then. If the function failes the program quits.

```
CInput::CInput(void)
{
    m_pDIObject = NULL;

    if(!InitDirectInput()) g_App.SetD3DStatus(false); //init DirectInput
} //CInput
```

The destructor is used to clean-up. If the pointer of the main object is not `NULL`, we release it to avoid memory leaks. Here the memory for the input devices will be released, too.

```
CInput::~~CInput(void)
{
```

```

if(m_pDIObject != NULL)
{
    m_pDIObject->Release(); //release main object
    m_pDIObject = NULL;
}
} //~CInput

```

Here we have the initialisation function. We create the main object with the function `DirectInput8Create()`. The first parameter is the instance of the application, which is easily accessible through `GetModuleHandle()`. The second is the direct input version, which is always set to `DIRECTINPUT_VERSION`. This is a constant defined in the `DirectInput` header, which always represents the actual version. The third parameter is the identifier of the interface and must set to `IID_IDirectInput8`, because we want to use `DirectInput 8.0`. Then we give over the pointer to the main object. The last is uninteresting so we set it to `NULL`. If the function fails we return false and create a message box to inform the user. Otherwise we return true.

```

bool CInput::InitDirectInput(void)
{
    if(FAILED(DirectInput8Create(GetModuleHandle(NULL),
                                DIRECTINPUT_VERSION,
                                IID_IDirectInput8,
                                (void**)&m_pDIObject,
                                NULL)))
    {
        MessageBox(g_App.GetWindowHandle,
                  "DirectInput8Create() failed!",
                  "InitDirectInput()",
                  MB_OK);
        return false;
    }

    return true;
} //InitDirectInput

```

main.h

Here we only have to include the `DirectInput` header and the input class header. Of course, we change the title and finally declare the pointer to the `CInput` pointer as `extern` to be able to use it in other files.

```

//includes
#include<windows.h>
#include<d3d9.h>
#include<d3dx9.h>
#include<dinput.h> //DirectInput header (NEW)

#include"cappplication.h"
#include"cinput.h" //input class header (NEW)

//constants
#define TITLE "DInput Tut 01: Init DirectInput"

//globals
extern CApplication g_App;
extern CInput* g_pInput; //extern input object (NEW)

```

main.cpp

In this file we add the global input class pointer. We use a pointer here, because for mouse support we need to know the screen resolution. This information is only available after the initialisation of Direct3D.

```
//globals
CApplication g_App;
CInput*      g_pInput; //global input object (NEW)
```

For now we dynamically create an object with `new` and simply `delete` it at the end.

```
g_pInput = new CInput;

delete g_pInput;
```

This is the input class for the moment. In the next tutorials we add keyboard, mouse and joystick support to this class.