CooperativeLevelFlags.Background

Article11/06/2009

Capturing device input using Microsoft DirectInput can be done in three simple steps. First you create a Device object that represents the input device you wish to capture. After you create the device object, you do any necessary configuration, such as calling the SetCooperativeLevel method. Once you have configured the device you can start checking the state of the device in your application.

- Creating DirectInput Device Objects
- Configuring DirectInput Device Objects
- Capturing from DirectInput Device Objects

Creating DirectInput Device Objects

As discussed in Enumerating DirectInput Devices, there are many ways to find out what input devices are currently attached to the user's computer. More often than not, you will have to capture input from more than one device.

The following C# example code illustrates how to create three different Device objects: one for the default keyboard on the user's computer, one for the default mouse, and one for a joystick.

```
[C#]
private void InitDevices()
{
    //create keyboard device.
    keyboard = new Device(SystemGuid.Keyboard);
    if(keyboard == null) throw new Exception("No keyboard found.");

    //create mouse device.
    mouse = new Device(SystemGuid.Mouse);
    if(mouse == null)
    {
        throw new Exception("No mouse found.");
    }

    //create joystick device.
    foreach(
        DeviceInstance di in
        Manager.GetDevices(
            DeviceClass.GameControl,
            EnumDevicesFlags.AttachedOnly))
    {
```

```
            joystick = new Device(di.InstanceGuid);
            break;
        }

        if(joystick == null)
        {
            //Throw exception if joystick not found.
            throw new Exception("No joystick found.");
        }
    }
```

Configuring DirectInput Device Objects

Because the input devices you capture from are a shared system resource, you have to configure them to cooperate with other system resources. You do this by calling the SetCooperativeLevel method of the Device objects you created with a combination of flags from the CooperativeLevelFlags enumeration.

CooperativeLevelFlags

Flag

Description

CooperativeLevelFlags.Background

This device can be used in the background and can be acquired at any time, even if the application window is not active.

CooperativeLevelFlags.Foreground

This device can only be used when the application window is in the foreground. When the application window loses focus, the device will no linger be acquired.

CooperativeLevelFlags.Exclusive

This device requires exclusive access. No other application can acquire this device. For security reasons, this flag cannot be used in conjunction with the background flag on certain devices such as a keyboard and mouse.

CooperativeLevelFlags.NonExclusive

This device can be shared with any other application that does not require exclusive access.

CooperativeLevelFlags.NoWindowsKey

This flag disables the window's key.

Another configuration task is setting the range of a joystick axis. This allows you to set a numeric range to the complete distance traveled on a joystick axis. Even if the joystick you are using does not support the resolution of the range you assign, DirectInput will translate the joystick's resolution into the range you assign.

The following C# example code demonstrates how to configure the devices created in the previous sample.

```
[C#]
//Set mouse axis mode absolute.
mouse.Properties.AxisModeAbsolute = true;

//Set joystick axis ranges.
foreach(DeviceObjectInstance doi in joystick.Objects)
{
    if((doi.ObjectId & (int)DeviceObjectTypeFlags.Axis) != 0)
    {
        joystick.Properties.SetRange(
            ParameterHow.ById,
            doi.ObjectId,
            new InputRange(-5000,5000));
    }
}

//Set joystick axis mode absolute.
joystick.Properties.AxisModeAbsolute = true;

//set cooperative level.
keyboard.SetCooperativeLevel(
    this,
    CooperativeLevelFlags.NonExclusive |
    CooperativeLevelFlags.Background);

mouse.SetCooperativeLevel(
    this,
    CooperativeLevelFlags.NonExclusive |
    CooperativeLevelFlags.Background);

joystick.SetCooperativeLevel(
    this,
    CooperativeLevelFlags.NonExclusive |
    CooperativeLevelFlags.Background);

//Acquire devices for capturing.
keyboard.Acquire();
mouse.Acquire();
joystick.Acquire();
```

Capturing from DirectInput Device Objects

Normally you would want to capture the device object's state at a certain point of your application.

The C# example code below contains methods for capturing device input from the devices created in the previous examples, and updating a label control. They can be called by an application whenever you need to capture DirectInput device input.

```
[C#]
private void UpdateKeyboard()
{
    string info = "Keyboard: ";

    //Capture pressed keys.
    foreach(Key k in keyboard.GetPressedKeys())
    {
        info += k.ToString() + " ";
    }
    lbKeyboard.Text = info;
}

private void UpdateMouse()
{
    string info = "Mouse: ";

    //Get Mouse State.
    MouseState state = mouse.CurrentMouseState;

    //Capture Position.
    info += "X:" + state.X + " ";
    info += "Y:" + state.Y + " ";
    info += "Z:" + state.Z + " ";

    //Capture Buttons.
    byte[] buttons = state.GetMouseButtons();
    for(int i = 0; i < buttons.Length; i++)
    {
        if(buttons[i] != 0)
        {
            info += "Button:" + i + " ";
        }
    }

    lbMouse.Text = info;
}

private void UpdateJoystick()
{
    string info = "Joystick: ";
```

```
    //Get Mouse State.
    JoystickState state = joystick.CurrentJoystickState;

    //Capture Position.
    info += "X:" + state.X + " ";
    info += "Y:" + state.Y + " ";
    info += "Z:" + state.Z + " ";

    //Capture Buttons.
    byte[] buttons = state.GetButtons();
    for(int i = 0; i < buttons.Length; i++)
    {
        if(buttons[i] != 0)
        {
            info += "Button:" + i + " ";
        }
    }

    lbJoystick.Text = info;
}
```