

Joystick

Hamed Sabri

September 16, 2006

Hamed.Sabri@gmail.com

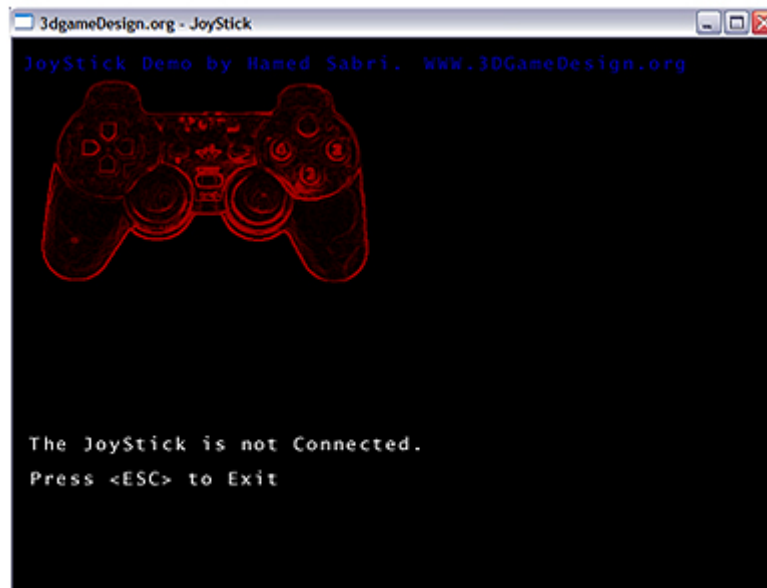
Introduction:

A **joystick** is a personal computer peripheral or general control device consisting of a handheld stick that pivots about one end and transmits its angle in two or three dimensions to a computer. Most joysticks are two-dimensional, having two axes of movement (similar to a mouse), but three-dimensional joysticks do exist. A joystick is generally configured so that moving the stick left or right signals movement along the X axis, and moving it forward (up) or back (down) signals movement along the Y axis. In joysticks that are configured for three-dimensional movement, twisting the stick left (counter-clockwise) or right (clockwise) signals movement along the Z axis. These three axis - X Y and Z - are, in relation to an aircraft, roll, pitch, and yaw. Joysticks are often used to control games, and usually have one or more push-buttons whose state can also be read by the computer. Most I/O interface cards for PCs have a joystick (game control) port. Joysticks were popular throughout the mid-1990s for playing the Descent series and flight-simulators, although they have declined with the rise of first-person shooters which instead promote the mouse and keyboard. Modern joysticks (as of 2003) mostly use a USB interface for connection to the PC. The term joystick has become a synonym for game controllers that can be connected to the computer since the computer defines the input as a "joystick input". Apart for controlling games, joysticks are also used for controlling machines such as elevators, cranes, trucks, powered wheelchairs and some zero turning radius lawn mowers. ([Wikipedia](#))

Let's start with the fun part. I assume that you have the basic knowledge of the programming with OpenGL and C++. Most of the description in this tutorial is from "*Beginning Game Programming*" by *Micheal Morrison*. I used it because it is nice piece of work, and can be understood very easily. Also, I used some graphic interface, so you can get the idea how it is done in OpenGL. The following pictures show the result when you run the program.



When the Joystick is connected.



When the Joystick is not connected.

STEP 1:

The first step in handling joystick input is checking to see if a joystick driver is installed and available on the computer system. Without the proper hardware drive in place, a physical joystick device is no good. This is made possible by a call to a win32 API function called **joyGetNumDevs()**. The **joyGetNumDevs()** tells you how many joysticks are capable of being

used on the computer system. The following is an example of how you might call the joyGetNumDevs() function to determine the number of joysticks available for use on the system:

```
UINT uiNumJoysticks = joyGetNumDevs();
```

To see if a real joystick is actually plugged in and ready to use, you call the joyGetPos() function, which provides a lot of information about a joystick. You must pass this function an ID that identifies the joystick you're using. Standard joystick ID's include JOYSTICKID1, JOYSTICKID2, and so on. So, to check for the presence of a single joystick, you can use code like this.

```
JOYINFO joy_info;  
  
if(joyGetPos(JOYSTICKID1, &joy_info) != JOYERR_UNPLUGGED)  
    joy_ID = JOYSTICKID1; // make sure the joystick is attached
```

The status of the joystick is stored in the JOYINFO structure, which is defined as follows:

```
typedef struct{  
    UNITwXpos;  
    UNITwYpos;  
    UNITwZpos;  
    UNITwButtons;  
}JOYINFO;
```

The first three members of the structure (wXpos, wYpos, wZpos) indicate the position of the joystick with respect to each of these axes. The final member, wButtons, indicates the state of the joystick buttons. The wButtons member supports up to four buttons, as indicated by the following constants:

```
joy_but1, joy_but2, joy_but3, joy_but4;
```

STEP 2:

The next step is to find out the minimum and maximum value used for each axis of movement that you are interested in checking. For example, if you want to see if the joystick has been moved left, you first need to find out the range of value for the x axis of the joystick. You can determine the ranges of joystick axes by calling the Win32 **JoyGetDevCaps()** function. This function fills a **JOYCAPS** structure with more information about a joystick than you'll probably ever want to know.

```

typedef struct {
    WORD wMid;
    WORD wPid;
    CHAR szPname[MAXPNAMELEN];
    UINT wXmin;
    UINT wXmax;
    UINT wYmin;
    UINT wYmax;
    UINT wZmin;
    UINT wZmax;
    UINT wNumButtons;
    UINT wPeriodMin;
    UINT wPeriodMax;
    UINT wRmin;
    UINT wRmax;
    UINT wUmin;
    UINT wUmax;
    UINT wVmin;
    UINT wVmax;
    UINT wCaps;
    UINT wMaxAxes;
    UINT wNumAxes;
    UINT wMaxButtons;
    CHAR szRegKey[MAXPNAMELEN];
    CHAR szOEMVxD[MAX_JOYSTICKOEMVXDNAME];
} JOYCAPS;

```

The following is a code snippet that determines the center point of the x,y,z axis, which reveals the ranges of each:

```

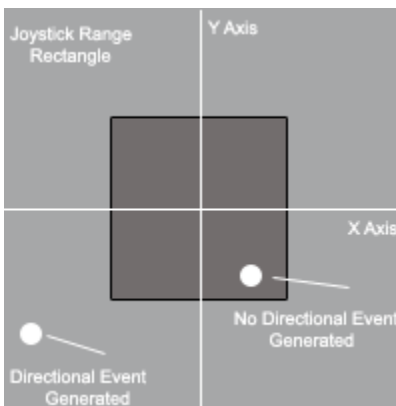
joyGetDevCaps(joy_ID, &joy_caps, sizeof(JOYCAPS));
joy_xcenter = ((DWORD)joy_caps.wXmin + joy_caps.wXmax) / 2;
joy_ycenter = ((DWORD)joy_caps.wYmin + joy_caps.wYmax) / 2;
joy_zcenter = ((DWORD)joy_caps.wZmin + joy_caps.wZmax) / 2;

```

STEP 3:

The next step is to find out the trip rectangle for the joystick. Trip rectangle is an area that determines how far the joystick handle must move in order for it to count as a directional

event(up,down,left,right ,or a combination).The purpose of the trip rectangle is to only cause joystick movement events to be generated if the handle moves a certain minimum distance.



```
RECT    joy_trip;
joy_trip.left    = (joy_caps.wXmin + (WORD)joy_xcenter) / 2;
joy_trip.right   = (joy_caps.wXmax + (WORD)joy_xcenter) / 2;
joy_trip.top     = (joy_caps.wYmin + (WORD)joy_ycenter) / 2;
joy_trip.bottom  = (joy_caps.wYmax + (WORD)joy_ycenter) / 2;
```

STEP 4:

The next step is to capture the joystick. This means that the joystick is only going to communicate with that program.

```
joySetCapture(hWnd, joy_ID, NULL, TRUE); // capture the joystick
```

The last step is to release the joystick so that is no longer captured.

```
joyReleaseCapture(joy_ID); // release the joystick
```

STEP 5:

The **CheckJoystick()** function compares different members of the JOYINFO() structure against the trip rectangle to see if the joystick movement qualifies as a directional movement. Also the joysticks buttons are checked and appropriate are set for those as well. The following are all the functions used in this tutorial.

```
bool InitJoystick();
void ReleaseJoystick();
void CheckJoystick();
```