## Using the Xbox-360 Marmalade Extension

This extension allows you to read input such as DPad, A/B/X/Y buttons and analog stick values from an Xbox-360 or compatible game-pad on PC. You can also use it with a PS3 Six-Axis controller connected via USB or a Bluetooth dongle. Note that for PS3 Six-Axis controllers to work, you will require a 3rd-party driver such as MotionJoy whereas the 360 controller is natively supported by Windows.

Using such input in your PC game can really make a huge difference to the game-play by providing the player with a rich and satisfying way to move around, fire, or otherwise control the player character. Xbox 360 controllers are very common and many PC game players will have bought an inexpensive one or will have received one with their Xbox 360.

You can set this extension up within a few minutes and get started using this wonderful input method by following the instructions below. My rationale for releasing this extension to the Marmalade community is simple – although I invested a lot of effort into making it, the wider benefit to the community will help my company too in the long run by helping ensure the success of Marmalade on the PC platform. 360 Controller support is a great selling point for your game that can be integrated literally in minutes and tested for the price of a cheap controller. When I was developing my game (Robots Can’t Jump) the first time I played it with the controller I was blown away by how much better and more natural the input felt than the keyboard or mouse.

I have included a complete bare-bones example that you should find it simple to copy the lines of code you need from to make this work in your project.

## Installation And Use

### Step Zero

Unzip the archive into **the same directory** as the rest of your project’s source files – this will make things easier later on when we are dealing with relative paths inside your project’s MKB.

### Step One

You will have to install the MS DDSK for the 360 extension to work – the extension uses the HID LIB files.   <http://www.microsoft.com/en-us/download/details.aspx?id=11800>

### Step Two

Open **GamePadExtension\source\windows\GamePad\_platform.cpp** in a text editor. At the top of the file there is a section with a bunch of **#pragma comment(lib, path)** lines. You will need to change these paths to match the path where you have installed the MS DDK in Step One.

### Step Three

Build the extension by right-clicking **GamePadExtension\GamePad.s4e** and selecting **Build Windows Extension**. A bunch of files will be created for you (mkb, mkf etc) and some directories.

### Step Four

Double-click **GamePadExtension\Gamepad\_windows.mkb** to open the GamePad project in Visual Studio. Build the project in both Debug and Release configurations – Debug is used for Debug builds of your app, Release for your Release build. If you don’t do both you’ll get weird linker errors later on.

### Step Five

In your own project’s MKB file, add the following to the ‘subprojects’ and ‘includepath’ sections, if you don’t already have them:

**subprojects**

**{**

**GamePadExtension\GamePad**

**}**

**includepath**

**{**

**GamePadExtension**

**}**

Note that you may need to change the paths to contain a relative path to wherever you have placed the GamePadExtension project after unzipping. In this example, the ‘GamePadExtension’ folder is in the same directory as your project’s MKB.

### Step Six

You do not have to call an Init method for the extension – this will be done for you automatically the first time you call any extension functions. The extension has a separate Windows window-procedure that needs constant updating to poll the input event messages from the controller. I recommend calling this method once per frame in your update loop. Here is an example:

**// Update the Xbox 360 game-pad, enabling input event processing etc**

**GamePad\_Update(0.0f);**

### Step Seven

You may want to call GamePad\_IsConnected() to check whether a gamepad is actually connected and ask the user to connect if they disconnect during game-play, etc. This is good practise and will make your game seem more professional.

### Step Eight

Now all you need to do is call the various GamePad\_Get\*() methods to read various stick and button statuses. Sticks use analog values ranging from -1 to 1 to indicate “fully moved”. The triggers use values from 0 to 1, where 1 is fully pressed. The buttons are just on or off. There are functions for every button, stick and trigger, including Start and Select. The special ‘Xbox’ button in the centre of the controller is not accessible as it has specia functions (XBLive, etc). Have fun!