Tutorial 6: Button detection

http://www.codemii.com/2008/09/14/tutorial-6-button-detection

In this Wii programming tutorial we will cover displaying an image as the mouse cursor and how to detect button clicking.

Firstly download this tutorial6-image which will contain the required files to get us started. This zip file also contains the JPEG code required to display images which you should have learnt about in the last tutorial. Extract this zip file in C:\devkitPro\examples\gamecube and open up tutorial6.pnproj.

As you should recall, we displayed a cursor which we could control on the gamecube by using the following code:

```
int cursor_x = 300;
int cursor_y = 250;

while(1) {

		PAD_ScanPads();

		if (PAD_StickY(0) > 18) {
			cursor_y--;
		}
		if (PAD_StickY(0) < -18) {
			cursor_y++;
		}
		if (PAD_StickX(0) > 18) {
			cursor_x++;
		}
		if (PAD_StickX(0) < -18) {
			cursor_x--;
		}

		VIDEO_ClearFrameBuffer (rmode, xfb, COLOR_BLACK);

		DrawBox (cursor_x, cursor_y, cursor_x + 1, cursor_y + 1, COLOR_WHITE);

		VIDEO_WaitVSync();

}
```

So all that's required for us to do is change DrawBox to use display_jpeg instead.

```
display_jpeg(about, cursor_x, cursor_y);
```

Compile your source and there we have it; your cursor is now showing up as an image.

Detecting button clicking

In order to detect button clicking we have to know our button's four co-ordinates; x1, x2, y1 and y2. The simplest way to figure this out is to just print the location of your cursor to the screen when you press A so you can see the x and y co-ordinates as when you are displaying an image it's not the same as the x and y of the cursor. I know that there has to be a better way for doing this, but I'm just showing you what worked for me.

Put change display_jpeg back to DrawBox as we'll need to find the x and y co-ordinates.

You can then add the following code to print out the x and y co-ordinates of the cursor:

```
u16 buttonsDown = PAD_ButtonsDown(0);

if( buttonsDown & PAD_BUTTON_A ) {
		printf("x = %i. y = %i\n", cursor_x, cursor_y);
		usleep(500000);
}
```

We have used printf before, which prints something to the screen but in this case we are using printf to print out two integers; x and y. The way the statement works is we say to printf that we will be passing an integer in the first argument which is denoted by %i. After we have finished the text to print (" "), we pass the first argument; cursor_x which is an integer. Printf grabs this integer and displays it when it prints "x = 100". You can print a double (%d), long (%l), characters (%c) and so on.

Usleep as the name suggests, makes the whole application pause for a specified amount of time. The argument usleep takes is a integer and is in microseconds. 500000 would be 500 milliseconds which is ½ of a second.

Before using usleep you need to include

```
#include <unistd.h>
```

in your headers. Unistd includes miscellaneous functions. If you didn't include unistd, it would still compile but throw an error saying it can't find the usleep function, but lucky for us it would still work.

So now if we compile and run our code (tutorial6-mouse-image), we can move the cursor around the about image. What we need to do is find the x, y co-ordinates of the top left and bottom right of the image. So if you were to go ahead and do this yourself you would find that they would roughly be 202, 120 (top left) and 254, 141 (bottom right).

We have our image's co-ordinates and now all that's left to do is produce code that will identify if our cursor is in our images co-ordinates. Something as found below works nicely:

```
if (cursor_x >= 202 && cursor_x <= 254 && cursor_y >= 120 && cursor_y <= 141) {
                printf("Button pressed\n");
}
```

Place the above code in the if (buttonsDown & PAD_BUTTON_A) { statement and recompile your code (tutorial6-button-detection). Once you've run it with gcube, try pressing the A button (Q key) around the button and inside the button. You'll see it works quite well and correctly detects when the button is pressed.