Tutorial 5: JPEG images

http://www.codemii.com/2008/09/07/tutorial-5-jpeg-images/

This fifth Wii programming tutorial will cover how to display JPEG images. As usual, I'm assuming you've read all the other tutorials.

Firstly download this tutorial5-blank which will contain the required files to get us started. Extract this zip file in C:\devkitPro\examples\gamecube. Open up tutorial5.pnproj and then click on main.c to show the source code.

Included in the download are two files, one called picture.s and a JPEG picture. The picture.s file defines all our variables for the picture and the link to the picture itself. If you open the image file "about.jpg" you'll see the image with a black background. It's important to remember that JPEGs aren't transparent like PNGs are.

The next step needed is to download this libogc_jpeg zip file which contains the required JPEG library. You'll need to extract it to C:\devkitPro\devkitPPC\powerpc-gekko and let it overwrite any directories.

Now we need to modify our makefile to add in the JPEG library:

```
LIBS    :=      -logc -lm -ljpeg
```

We can now focus on editing the main.c file. Firstly we'll need to include the jpeg header file so we can access the jpeg functions.

```
#include <jpeg/jpgogc.h>
```

Two variables need to be defined are piclength and picdata which are both found in the picture.s file. We define these two variables as exern which tells the compiler that we have defined these two variables elsewhere (picture.s) and these variables will be able to be accessed globally.

```
extern char picdata[];
extern int  piclength;
```

The next thing we add is a jpeg function which I have hacked together:

```
void display_jpeg(JPEGIMG jpeg, int x, int y) {

        unsigned int *jpegout = (unsigned int *) jpeg.outbuffer;

        int i,j;
        int height = jpeg.height;
        int width = jpeg.width/2;
                for(i=0;i<=width;i++)
                        for(j=0;j<=height-2;j++)
                                xfb[(i+x)+320*(j+16+y)]=jpegout[i+width*j];

        free(jpeg.outbuffer);

}
```

The above function takes a JPEGIMG structured variable and two integers which will be used to determine where on the screen we want our image to be displayed.

The first line in the function is related to how the jpeg library works. Later on you'll see that we decompress the jpeg images and when doing so, the jpeg.outbuffer is where the image is stored.

The next few lines define some local variables for the height and width and then copy the jpeg image to out video frame buffer (the screen) to the co-ordinates that we have defined in as x and y. The last line in the function ensures that we clear the space in memory that we made a local copy of the jpeg image (unsigned int *jpegout = (unsigned int *) jpeg.outbuffer;)

We can now start adding jpeg functions the main() function. First things first we define the variable to use that will use the JPEGIMG data structure.

```
JPEGIMG about;
```

After that we assign memory to our "about" variable.

```
memset(&about, 0, sizeof(JPEGIMG));
```

According to our picture.s file, it contains two variables; the picture data and the length of the picture. We define our "about" variable inbuffer and inbufferlength with both the global variables.

```
about.inbuffer = picdata;
about.inbufferlength = piclength;
```

We then need to decompress the jpeg, by passing the "about" variable.

```
JPEG_Decompress(&about);
```

Our last step is to run the display_jpeg function and pass our variable “about” and any x and y co-ordinates we want.

```
display_jpeg(about, 60, 100);
```

Our main function now looks like:

```
int main() {

	JPEGIMG about;

	memset(&about, 0, sizeof(JPEGIMG));

	about.inbuffer = picdata;
	about.inbufferlength = piclength;

	JPEG_Decompress(&about);

	Initialise();

	display_jpeg(about, 60, 100);

	return 0;
}
```

tutorial5-jpeg is what our source now looks like. Compile the source and run tutorial5.dol with gcube. You’ll see the image displayed on the screen. I won’t convert this to the Wii as there won’t be any difference between them.