

# Donkey Kong Country Sprites

This is a guide on how to manually edit the sprite graphics in Donkey Kong Country. No previous knowledge required but knowing how to use a Hex editor is recommended. I use Translhexion.

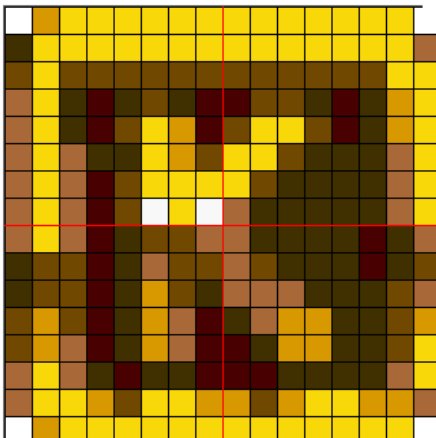
I will try to start off with very simple examples with images and work up to something a little more complicated. There is a lot of redundancy in this document. If you are looking for something more in depth check out this awesome Tutorial by georgjz here! <https://georgjz.github.io/snesaa01/> Most of the info (and some images) I got from there!

Another note... This is not the best way of making/editing your own sprites. There are tools such as YY-CHR <https://www.smwcentral.net/?p=section&a=details&id=4642> that automate things for you but this guide is meant to give you a better understanding on how the SNES handles sprites.

## Section One (What are Sprites?)

The characters and objects in an SNES game are called sprites. Sprites are made up of 1 or more tiles and each tile is 8x8 pixels. A single 8x8 4bpp tile is 32 bytes.

Below is a sprite made up of four 8x8 tiles.



## Section Two (Colour Indexing)

The SNES uses Colour Indexing to do its colour. This means the index number of a colour references a colour inside a palette.

The below image is in 1bpp (Bits Per Pixel) colour format. Since it is only one bit (0 or 1) only two colours are available.

Sprite	Color Indices	Mem.	Palette	
	0 0 0 1 1 0 0 0	\$18	0	
	0 0 1 1 1 1 0 0	\$3c	1	
	0 1 1 1 1 1 1 0	\$7e		
	1 1 0 1 1 0 1 1	\$db		
	1 1 1 1 1 1 1 1	\$ff		
	0 0 1 0 0 1 0 0	\$24		
	0 1 0 1 1 0 1 0	\$5a		
	1 0 0 0 0 0 0 1	\$81		

### Section Three (Colour Formats)

The Donkey Kong Country game uses the 4bpp (Bits Per Pixel) colour format for its sprites. The SNES uses 2 colour formats Intertwined and Planar. Donkey Kong Country uses the Intertwined format.

The simplest colour format as shown above is 1bpp or 1 bit per pixel. This means each pixel can contain one bit or two colours. Colour 0 or colour 1. Colour 0 is always transparent

Below is an example of a 2bpp sprite. 4 colours are now available.

Sprite



Color Indices

0	1	1	1	1	1	1	1	0	
1	1	1	1	1	1	1	1	1	
1	2	2	2	2	2	2	2	1	
2	2	2	2	2	2	2	2	2	
2	3	1	3	3	1	3	2		
2	3	2	3	3	2	3	2		
3	3	3	3	3	3	3	3		
0	3	3	2	2	3	3	0		

Palette

0	
1	
2	
3	

### Section Four (Bitplanes and intertwined stored colour)

As an example a 2bpp sprite, we need to use Bitplanes to store the colour data.

bitplane 1 (MSB)

1	1	1	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0
1	1	1	1	1	0	0	1	
1	1	1	0	1	1	1	1	
1	1	1	1	1	0	0	0	
1	1	1	0	0	0	0	0	
1	1	1	0	0	0	0	0	
1	1	1	0	0	1	1	1	
1	1	1	0	0	0	0	0	


bitplane 0 (LSB)

0	0	1	1	1	1	1	1	
0	0	0	1	1	1	1	1	
0	0	1	0	0	1	1	1	
0	0	1	1	0	0	0	0	
0	0	1	0	0	1	1	1	
0	0	1	1	1	1	1	1	
0	0	1	1	1	1	1	1	
1	1	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	

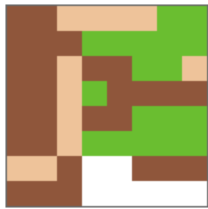
Palette

0	
1	
2	
3	

Color Indices



Sprite



bitplane 1 (MSB)

1	1	1	1	1	1	1	1	\$ff
1	1	1	0	0	0	0	0	\$e0
1	1	1	1	1	0	0	1	\$f9
1	1	1	0	1	1	1	1	\$ef
1	1	1	1	1	0	0	0	\$f8
1	1	1	0	0	0	0	0	\$e0
1	1	1	0	0	1	1	1	\$e7
1	1	1	0	0	0	0	0	\$e0

bitplane 0 (LSB)

0	0	1	1	1	1	1	1	\$3f
0	0	0	1	1	1	1	1	\$1f
0	0	1	0	0	1	1	1	\$27
0	0	1	1	0	0	0	0	\$30
0	0	1	0	0	1	1	1	\$27
0	0	1	1	1	1	1	1	\$3f
1	1	0	0	0	0	0	0	\$c0
0	0	0	0	0	0	0	0	\$00

Planar

\$3f	row 0
\$1f	
\$27	
\$30	
\$27	
\$3f	
\$c0	
\$00	row 7
\$ff	row 0
\$e0	
\$f9	
\$ef	
\$f8	
\$e0	
\$e7	
\$e0	row 7

Intertwined

\$3f	row 0
\$ff	
\$1f	
\$27	
\$e0	
\$f9	
\$30	
\$ef	
\$27	
\$f8	
\$3f	
\$e0	
\$c0	
\$e7	
\$00	row 7
\$e0	

This is because 2 bits (4 colours) need to be stored for each pixel. Each byte in a Bitplane refers to 1 row of a tile. A single Bitplane holds 1 bit of the colour index of the corresponding pixel. To get the colour index of a certain pixel, each bit in a Bitplane are combined. If you wanted to get the colour index of the Top most Left pixel of the tile we need to combine each bitplane. In this case a 2bit number because we are using the 2bpp colour format.

What is Intertwined? It means each pair of bytes (one from each bitplane) is stored from the top most row to the bottom. The lower bitplane is stored first.



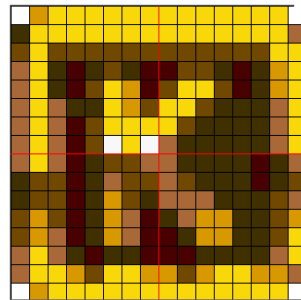
### Section Five (Palettes)

The SNES uses the BGR555 colour format. The SNES stores the data in reverse. Donkey Kong Country Sprites are in 4bpp colour format which means it can use up to a 16 colour palette. Colour 0 is always transparent.

## Tutorial

This is a tutorial on how to edit a single sprite in Donkey Kong Country. A Step by Step guide. I know it doesn't look very interesting but its supposed to be simple and straightforward.

You will be editing the below sprite.



To look like something like this.



### Part One (Donkey Kong Country Sprite Header Format)

Donkey Kong Country has its own Header format of 8 bytes followed by a 2 byte coordinate of each sprite. Below is a summary of the header format.

#### Sprite Header Format (Credit rainbowsprinklez from DKC-Atlas Forums)

- Byte 0 is number of 2x2 chars
- Byte 1 is number of 1x1 chars in group 1
- Byte 2 is relative position of first 1x1 char of group 1
- Byte 3 is number of 1x1 chars in group 2
- Byte 4 is position of group 2
- Byte 5 is number of chars in dma group 1
- Byte 6 is where to place dma group 2 (0 if none)
- Byte 7 is number of chars in dma group 2 (0 if none)

These 8 bytes are followed by 2 bytes each representing bytes 0, 1, and 3 of the header. Those 2 bytes are the X and Y coordinate of each graphic.

Below is the header of the Letter K sprite.



# of 2x2 chars	<input type="text" value="1"/>
# of 1x1 chars (group 1)	<input type="text" value="0"/>
1x1 offset (group 1)	<input type="text" value="0"/>
# of 1x1 chars (group 2)	<input type="text" value="0"/>
1x1 offset (group 2)	<input type="text" value="0"/>
Size to send to vram shifted 5 times (dma group 1)	<input type="text" value="2"/>
Offset in VRAM dma group 2 (0 if none)	<input type="text" value="10"/>
# of chars in dma group 2 (0 if none)	<input type="text" value="2"/>

The address of the sprite header is located at **0xF47A7D** (PC Address is **347A7D**) Subtract C00000 to get the PC Address.

Below is the Header data in my hex editor. 8 bytes followed by 2 byte coordinates. Total 10 bytes.

```
00347A54 00 4F 00 43 00 79 00 7F 00 DB A5 3A C5 7D C3 9C A2 7D 42 0F F2 F9 01 FE 00 40 04 00 04 00 00 41 00 81 00 01
00347A78 00 FE 00 FE 00 01 00 00 00 00 02 10 02 78 73 3F 00 FF 00 40 BF EA 95 E4 99 FC A1 E7 B8 E2 BD 7F 00 7F 00 40
00347A9C 00 40 11 46 11 46 00 47 10 47 10 FE 00 FF 01 03 FC 15 E8 65 98 DF 22 BF 42 FF 82 FE 00 FE 00 03 00 03 88 63
```

#### Sprite Graphic Data (All 4 tiles of 32bytes each = total of 128bytes)

```
00347A54 00 4F 00 43 00 79 00 7F 00 DB A5 3A C5 7D C3 9C A2 7D 42 0F F2 F9 01 FE 00 40 04 00 04 00 00 41 00 81 00 01
00347A78 00 FE 00 FE 00 01 00 00 00 00 02 10 02 78 73 3F 00 FF 00 40 BF EA 95 E4 99 FC A1 E7 B8 E2 BD 7F 00 7F 00 40
00347A9C 00 40 11 46 11 46 00 47 10 47 10 FE 00 FF 01 03 FC 15 E8 65 98 DF 22 BF 42 FF 82 FE 00 FE 00 03 00 03 88 63
00347AC0 08 C1 00 81 00 01 00 E9 B7 8C 77 89 72 0A B3 2A B3 F6 A9 E6 88 3F 00 40 10 00 10 04 10 44 11 44 11 40 09 77
00347AE4 00 7F 00 FB 85 BA C5 FD E3 CC 42 4D 82 3F C2 19 41 FE 00 00 04 00 04 00 00 31 00 31 80 01 C0 BE 00 FE 00 01
00347B08 00 00 00 00 02 10 02 78 73 3F 00 FF 00 40 BF E5 9A E2 9D FF A1 F3 AD F5 AB 7F 00 7F 00 40 00 40 18 42 18 42
```

#### Sprite Graphic Data (First Tile. Upper Left)

```
00347A74 00 81 00 01 00 FE 00 FE 00 01 00 00 00 00 02 10 02 78 73 3F
00347A88 00 FF 00 40 BF EA 95 E4 99 FC A1 E7 B8 E2 BD 7F 00 7F 00 40
00347A9C 00 40 11 46 11 46 00 47 10 47 10 FE 00 FF 01 03 FC 15 E8 65
00347AB0 98 DF 22 BF 42 FF 82 FE 00 FE 00 03 00 03 88 63 08 C1 00 81
```

## Part Two (From the beginning)

Lets start from the beginning by making all pixels transparent (writing zeros) in the hex editor so we can better see what we are doing. A value of zero in a sprite is always transparent.

Your hex editor should look something like this.

```
00347A30 | 00 03 00 F3 | 08 C3 18 C1 | 00 41 00 41 | 80 FE A1 9E | 61 9E 61 1D | A2 33 A0 EC | B0 F0 86 3F | 00 40 01 00 | 01 00 01 4C
00347A54 | 00 4F 00 43 | 00 79 00 7F | 00 DB A5 3A | C5 7D C3 9C | A2 7D 42 0F | F2 F9 01 FE | 00 40 04 00 | 04 00 00 41 | 00 81 00 01
00347A78 | 00 FE 00 FE | 00 01 00 00 | 00 00 02 10 | 02 78 73 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00
00347A9C | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00
00347AC0 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00
00347AE4 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 01
00347B08 | 00 00 00 00 | 02 10 02 78 | 73 3F 00 FF | 00 40 BF E5 | 9A E2 9D FF | A1 F3 AD F5 | AB 7F 00 7F | 00 40 00 40 | 18 42 18 42
```

And the sprite in game will look like this.









Image of the first tile of sprite changed to RED.



### Part Five (The Next Tile)

We will do another example by changing the second tile (Upper Right) to green which has the index value of 14 (0xE)



Converting 0xE to Binary is 1110

```

1 1 1 0
1 1 1 0
1 1 1 0
1 1 1 0
1 1 1 0
1 1 1 0
1 1 1 0
1 1 1 0
    
```

All eight rows of Bitplane 0 add up to 0x00.  
 All eight rows of Bitplane 1 add up to 0xFF.  
 All eight rows of Bitplane 2 and 3 add up to 0xFF.

Remember the colour format is intertwined

So when writing the byte values to file you need to alternate between the two bitplanes 0 and 1. Then write the Bitplane values of 2 and 3. For example... To write the first two bitplanes to file.

0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF

```

00347A6C | 00 40 04 00 | 04 00 00 41 | 00 81 00 01 | 00 FE 00 FE | 00 01 00 00 | 00 00 02 10 | 02 78 73 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 |
00347A96 | 00 FF FF FF | FF FF FF FF | FF FF FF FF | FF FF FF FF | FF FF FF FF | FF 00 FF 00 | FF 00 FF 00 | FF 00 FF 00 | FF 00 FF 00 | FF 00 FF 00 | FF 00 FF 00 | FF 00 FF 00 | FF 00 FF 00 | FF 00 FF 00 | FF 00 FF 00 | FF 00 FF 00 |
00347AC0 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 |
    
```

That's a total of 16 bytes (8 bytes x 2 Bitplanes)

Next write the bytes of bitplane's 2 and 3 to file.

0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF

That's a total of 16 bytes (8 bytes x 2 Bitplanes)

All the bitplanes add up to 32bytes.

The Finished Green Tile



### Part Six (The Bottom Left Tile)

Last we will change the rest of the tiles, the Bottom Left and Bottom Right.



The index number for the Colour Yellow is 5 or 0x05 in Hex.

Converting 0x05 to Binary is 0 1 0 1

0 1 0 1  
0 1 0 1  
0 1 0 1  
0 1 0 1  
0 1 0 1  
0 1 0 1  
0 1 0 1  
0 1 0 1

Converting Bitplane's 0 and 1

0xFF, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0x00

00347A70	04 00 00 41	00 81 00 01	00 FE 00 FE	00 01 00 00	00 00 02 10	02 78
00347A86	73 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 FF FF FF	FF FF
00347A9C	FF FF FF FF	FF FF FF FF	FF FF FF 00	FF 00 FF 00	FF 00 FF 00	FF 00
00347AB2	FF 00 FF 00	FF FF FF FF	FF FF FF FF	FF FF FF FF	FF FF FF FF	FF FF
00347AC8	00 FF 00 FF	00 FF 00 FF	00 FF 00 FF	00 FF 00 00	00 00 00 00	00 00
00347ADE	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00
00347AF4	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 01	00 00
00347B0A	00 00 02 10	02 78 73 3F	00 FF 00 40	BF E5 9A E2	9D FF A1 F3	AD F5

Converting Bitplane's 2 and 3

0xFF, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0xFF, 0x00

00347AA6	FF 00 FF 00	FF 00 FF 00	FF 00 FF 00	FF 00 FF 00	FF FF FF FF	FF FF FF FF	FF FF FF FF	FF FF FF FF	FF FF FF FF	FF FF
00347AC8	00 FF 00 FF	00 FF 00 FF	00 FF 00 FF	00 FF 00 FF	FF 00 FF 00 FF	FF 00 FF 00 FF	FF 00 FF 00 FF	FF 00 FF 00 FF	FF 00 FF 00	00 00 00
00347AEA	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 01 00 00	00 00

The Finished Yellow Tile.



### Part Seven (Last Tile)

I won't go into detail for this. You should get the idea by now!



White has the index value of 6 or 0x06 in hex. 0x06 in binary is 0 1 1 0.

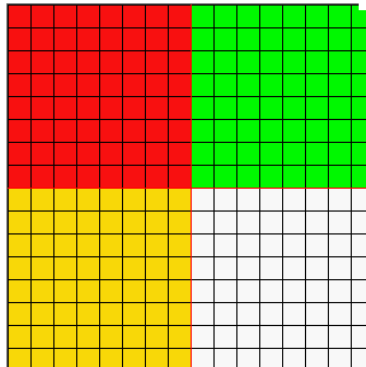


### Completed Sprite in Hex

```

00347A40 | 61 9E 61 1D | A2 33 A0 EC | B0 F0 86 3F | 00 40 01 00 | 01 00 01 4C | 00 4F 00 43 | 00 79 00 7F | 00 DB A5 3A | C5 7D
00347A62 | C3 9C A2 7D | 42 0F F2 F9 | 01 FE 00 40 | 04 00 04 00 | 00 41 00 81 | 00 01 00 FE | 00 FE 00 01 | 00 00 00 00 | 02 10
00347A84 | 02 78 73 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 FF | FF FF FF FF | FF FF FF FF | FF FF FF FF | FF FF
00347AA6 | FF 00 FF 00 | FF 00 FF 00 | FF 00 FF 00 | FF 00 FF 00 | FF FF FF FF | FF FF FF FF | FF FF FF FF | FF FF FF FF | FF FF
00347AC8 | 00 FF 00 FF | 00 FF 00 FF | 00 FF 00 FF | 00 FF 00 FF | 00 FF 00 FF | 00 FF 00 FF | 00 FF 00 FF | 00 FF 00 FF | 00 FF 00 FF
00347AEA | FF 00 FF 00 | FF 00 FF 00 | FF 00 FF 00 | FF 00 FF 00 | FF FF 00 FF | 00 FF 00 FF | 00 FF 00 FF | 00 FF 00 FF | 00 01 00 00 | 00 00
00347B0C | 02 10 02 78 | 73 3F 00 FF | 00 40 BF E5 | 9A E2 9D FF | A1 F3 AD F5 | AB 7F 00 7F | 00 40 00 40 | 18 42 18 42 | 00 42
    
```

Note- It may look one pixel off. Not sure why that is. I tried it with a different colour and it lined up perfectly! Also to the right is an image from rainbowsprinklez graphic editor and as you can see all 4 tiles line up perfectly!



See they line up....



The example above in Hex editor.

00347A48	B0	F0	86	3F	00	40	01	00	01	00	01	4C	00	4F	00	43	00	79	00	7F	00	DB	A5	3A	..
00347A60	C5	7D	C3	9C	A2	7D	42	0F	F2	F9	01	FE	00	40	04	00	04	00	00	41	00	81	00	01	
00347A78	00	FE	00	FE	00	01	00	00	00	00	02	10	02	78	73	00	00	00	00	00	00	00	00	00	00
00347A90	00	00	00	00	00	00	00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	00
00347AA8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	FF	00	FF	00	FF	00	FF	00	FF	
00347AC0	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	
00347AD8	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	00	FF	00	FF	00	FF	00	FF	00	
00347AF0	FF	00	FF	00	FF	00	FF	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	01
00347B08	00	00	00	00	02	10	02	78	73	3F	00	FF	00	40	BF	E5	9A	E2	9D	FF	A1	F3	AD	F5	
00347B20	AB	7F	00	7F	00	40	00	40	18	42	18	42	00	42	08	42	08	FE	00	FF	01	03	FC	05	

### Conclusion

That's it! You completed the tutorial and made your own sprite by hand!

I hope it wasn't too confusing for you. If you have any comments you can email me at [Cyclone.Chris@gmail.com](mailto:Cyclone.Chris@gmail.com) or visit the [DKC-Atlas.com](http://DKC-Atlas.com) forums.

### Credits

Cyclone (That's me who made this Doc ;) )

rainbowsprinklez (DKC-Atlas Forums)

georgjz from <https://georgjz.github.io/snesaa01/> (This is where I got most of the info for this Document.)

Kingizor (DKC-Atlas forums)

Matrizzel (DKC-Atlas forums)

**Last Updated November 24th 2022**