# Furcadia Color Code Format

Author: Aleksi Asikainen (sanct@furcadia.com)
Last Update: 2007-08-11
Scope: Description of the new Furcadia Color Code Format

## Introduction

Furcadia Color Code is an arbitrary length of ASCII characters used to describe the colors or the color slides used to recolor the avatar of a Furcadia player (often referred as “character”).

Furcadia Color Code is encoded in 8-bit ASCII format, in which the bytes must have values between 35 (‘#’) and 255. This follows Furcadia Base 220 encoding.

Decoded values can be retrieved by subtracting 35 from each byte.

| Chunk | Data Type | Description |
|---|---|---|
| Color Code Type | BYTE | Defines the format of the color code. Must be set to ’t’ (0x74), ’u’ (0x75), or ’v’ (0x76).<br><br>‘t’ – “old format code”<br><br>**Reserved for future:**<br>‘u’ – “new format code”<br>‘v’ – “new format code – all RGB” |
| Color Code Data | 13-30 BYTES | *See below.* |

## Old Format Code

| Chunk | Data Type | Description |
|---|---|---|
| Fur Color | BYTE | Defines a pre-set color slide for fur (0-24). |
| Markings Color | BYTE | Defines a pre-set color slide for markings (0-24). |
| Hair Color | BYTE | Defines a pre-set color slide for hair (0-44). |
| Eye Color | BYTE | Defines a pre-set color for eyes (0-29). |
| Badge Color | BYTE | Defines a pre-set color for badge (0-39). |
| Vest Color | BYTE | Defines a pre-set color slide for vest (0-29). |
| Bracer Color | BYTE | Defines a pre-set color slide for bracers (0-29). |
| Cape Color | BYTE | Defines a pre-set color slide for cape (0-29). |

| | | |
|---|---|---|
| Boot Color | BYTE | Defines a pre-set color slide for boots (0-29). |
| Trousers Color | BYTE | Defines a pre-set color slide for trousers (0-29). |
| Gender | BYTE | *May be unavailable.*<br><br>Avatar gender:<br>0 – female<br>1 – male<br>2 – unknown |
| Species | BYTE | *May be unavailable.*<br><br>Avatar species:<br>0 – rodent<br>1 – equine<br>2 – feline<br>3 – canine<br>4 – musteline<br>5 – lapine<br>6 – squirrel<br>7 – bovine |
| Special Avatar | BYTE | *May be unavailable.*<br>Contains information about the special avatar currently active on the character. If the character does not have an active special avatar, this value should be zero. |

## New Format Code

***This information is preliminary and subject to change. As of March 17, 2007, there has been no decision made when the new format code will be introduced. Furcadia does not currently support RGB color slides in color codes.***

In order to support user specified RGB slides, the color code may be encoded in the following format when necessary to relay RGB data. When the color code contains encoded data, *Color Code Type* is set to 'u' or 'v'.

If *Color Code Type* is set to 'u', the color code begins with a two bytes long bitmask. The color data follows in the order specified in Old Format Code. The data chunks are either one or three bytes long:

| Bitmask set or Byte 0 = 'u' (0x75) | Bitmask not set and Byte 0 = 't' (0x74) |
|---|---|
| RGBDATA (3 BYTES) | BYTE |

Each byte of the bitmask contains up to six bits of information. In general this information follows the order presented in Old Format Code, but one specific exception applies: Eye and badge colors are described by a single bit – If either color is RGB encoded then *both* will be RGB encoded.

| Bitmask | Bit | Description |
|---|---|---|
| A | 1 | Fur color encoding |
| A | 2 | Markings color encoding |
| A | 3 | Hair color encoding |
| A | 4 | Badge and eye color encoding |
| A | 5 | Vest color encoding |
| A | 6 | Bracers color encoding |
| B | 1 | Cape color encoding |
| B | 2 | Boot color encoding |
| B | 3 | Trousers color encoding |

After the color data, three bytes *may* follow, specifying the gender, species, and special avatar respectively. These bytes bear no difference to the last three bytes described in Old Format Code.

Pseudocode example for reading the color data follows:

```
if Byte 0 = ‘u’ then
         Byte: Bitmask A
         Byte: Bitmask B
end if


if ( Bitmask A Bit 1 is set ) or ( Byte 0 = ‘v’ ) then
         Byte: Color slider RGBDATA for color1: fur
         Byte: Color slider RGBDATA for color1: fur
         Byte: Color slider RGBDATA for color1: fur
else
         Byte: fur (see “old format code” table above)
end if


if ( Bitmask A Bit 2 is set ) or ( Byte 0 = ‘v’ ) then
         Byte: Color slider RGBDATA for color2: markings
         Byte: Color slider RGBDATA for color2: markings
         Byte: Color slider RGBDATA for color2: markings
else
         Byte: color2: markings (see “old format code” table above)
end if


if ( Bitmask A Bit 3 is set ) or ( Byte 0 = ‘v’ ) then
         Byte: Color slider RGBDATA for color3: hair
         Byte: Color slider RGBDATA for color3: hair
         Byte: Color slider RGBDATA for color3: hair
else
         Byte: color3: hair (see “old format code” table above)
```

```
end if


if ( Bitmask A Bit 4 is set ) or ( Byte 0 = ‘v’ ) then
         Byte: Color RGB for color 4 & 5: eyes, badge
         Byte: Color RGB for color 4 & 5: eyes, badge
         Byte: Color RGB for color 4 & 5: eyes, badge
else
         Byte: color4: eyes (see “old format code” table above)
         Byte: color5: badge (see “old format code” table above)
end if


if ( Bitmask A Bit 5 is set ) or ( Byte 0 = ‘v’ ) then
         Byte: Color slider RGBDATA for color6: vest
         Byte: Color slider RGBDATA for color6: vest
         Byte: Color slider RGBDATA for color6: vest
else
         Byte: color6: vest (see “old format code” table above)
end if


if ( Bitmask A Bit 6 is set ) or ( Byte 0 = ‘v’ ) then
         Byte: Color slider RGBDATA for color7: bracers
         Byte: Color slider RGBDATA for color7: bracers
         Byte: Color slider RGBDATA for color7: bracers
else
         Byte: color7: bracers (see “old format code” table above)
end if



if ( Bitmask B Bit 1 is set ) or ( Byte 0 = ‘v’ ) then
         Byte: Color slider RGBDATA for color8: cape
         Byte: Color slider RGBDATA for color8: cape
         Byte: Color slider RGBDATA for color8: cape
else
         Byte: color8: cape (see “old format code” table above)
end if


if ( Bitmask B Bit 2 is set ) or ( Byte 0 = ‘v’ ) then
         Byte: Color slider RGBDATA for color9: boots
         Byte: Color slider RGBDATA for color9: boots
         Byte: Color slider RGBDATA for color9: boots
else
         Byte: color9: boots (see “old format code” table above)
end if


if ( Bitmask B Bit 3 is set ) or ( Byte 0 = ‘v’ ) then
         Byte: Color slider RGBDATA for color10: trousers
         Byte: Color slider RGBDATA for color10: trousers
         Byte: Color slider RGBDATA for color10: trousers
else
         Byte: color10: trousers (see “old format code” table above)
```

```
end if


if( Bytes left >= 3 ) then
         Byte: Gender (see "old format code" table above)
         Byte: Species (see "old format code" table above)
         Byte: Reserved (see "old format code" table above)
end if
```

## RGB Encoding

Furcadia Color Code encodes two 24-bit RGB values into three bytes by decreasing the RGB data resolution heavily: Each red and blue value is scaled down to a value range of 0-14, and each green value is scaled down to a value range of 0-13.

| Color Component | Supported Range |
|---|---|
| Red | 0-14 |
| Green | 0-13 |
| Blue | 0-14 |

The algorithm for encoding the two RGB values is as follows:

```
u =   int( r1 / 255 * 14 ) +
      ( 15 * int( g1 / 255 * 13 ) +
        ( 14 * int( b1 / 255 * 14 ) +
        ( 15 * int( r2 / 255 * 14 ) +
        ( 15 * int( g1 / 255 * 13 ) +
        ( 14 * int( b2 / 255 * 14 )
      ) ) ) ) )
```

*u* is then converted into three "base 220" bytes. It should be noted that this example already the "base 220 base value" (35) to each byte:

```
for i = 0 to 2
      BYTE u2 = ( u modulo 220 ) + 35
      u = int( u / 220 )
      write u2
next i
```

## RGB Decoding

When decoding an RGB value, the encoding process is simply reversed. Please note that “double” below refers to the data type, not multiplication.

$$u = ( \text{byte1} - 35 ) + ( \text{byte2} - 35 ) * 220 + ( \text{byte3} - 35 ) * 220 * 220$$

$$r1 = ( u \% 15 ) / 14 * 255$$

$$g1 = ( ( u / 15 ) \% 14 ) / 13 * 255$$

$$b1 = ( ( u / ( 15 * 14 ) ) \% 15 ) / 14 * 255$$

$$r2 = ( ( u / ( 15 * 14 * 15 ) ) \% 15 ) / 14 * 255$$

$$g2 = ( ( u / ( 15 * 14 * 15 * 15 ) ) \% 14 ) / 13 * 255$$

$$b2 = ( ( u / ( 15 * 14 * 15 * 15 * 14 ) ) \% 15 ) / 14 * 255$$

Pseudocode follows:

```
r1 = double( u modulo 15 ) / 14.0 * 255.0
u = u / 15

g1 = double( u modulo 14 ) / 13.0 * 255.0
u = u / 14

b1 = double( u modulo 15 ) / 14.0 * 255.0
u = u / 15

r2 = double( u modulo 15 ) / 14.0 * 255.0
u = u / 15

g2 = double( u modulo 14 ) / 13.0 * 255.0
u = u / 14

b2 = double( u modulo 15 ) / 14.0 * 255.0
```

## Example Color Codes

```
t//&%#;;;;;$&#
```

```
74 2f 2f 26 25 23 3b 3b
3b 3b 3b 24 26 23
```

```
u:*6ôy6ôy6ôy7E6ôy16ôy6ôy6ôy$##
```

```
75 3a 2a 36 f4 79 36 f4
79 36 f4 79 37 45 36 f4
79 31 36 f4 79 36 f4 79
36 f4 79 24 23 23
```

```
v6x&÷IƒÌ…aþ±ÀÈ-\ ’`kµ…ÒLƒÓu®%(#
```

```
76 36 78 26 f7 49 83 cc
85 61 fe b1 c0 c8 ad 5c
a0 92 60 6b b5 85 d2 4c
83 d3 75 ae 25 28 23
```