Using Color In CircleMUD

Jeremy Elson

November 17, 2002

Abstract

This document is a description of how to write C code which displays messages to players in color in the CircleMUD game engine. Its intended audience is for Coders of CircleMUD.

CircleMUD allows you to create colorful messages by using ANSI control sequences. Each player may select what "level" of color he/she desires from the four levels "off," "sparse," "normal," and "complete." Each player can select his/her color level by using the `COLOR` command from within the MUD; you as the programmer must decide which messages will be colored for each of the color levels.

All files in which you wish to use color must have the line:

```
#include "screen.h"
```

This should be put in after all other includes in the beginning of the file.

There are 8 colors available – "normal," red, green, yellow, blue, magenta, cyan and white. They are accessible by sending control sequences as part of another string, for example:

```
sprintf(buf, "If you're %shappy%s and you know it clap "
             "%d of your hands.\n\r", x, y, num_of_hands);
send_to_char(buf, ch);
```

In this example, `x` and `y` are the "on" and "off" sequences for the color you want. There are 2 main series of color macros available for you to use (don't actually use "x" and "y," of course!): the K series and the CC series. The CC (Conditional Color) series is recommended for most general use.

The name of the actual sequence starts with the name of its series, plus a 3-letter color code, as follows:

Normal: NRM

Red: RED

Yellow: YEL

Green: GRN

Blue: BLU

Magenta: MAG

Cyan: CYN

White: WHT

For example, white in the K series is `KWHT`; blue in the CC series is `CCBLU()` (arguments defined below).

The K series requires no arguments, and is simply a macro to the ANSI color code. Therefore, if you use a K-series color code, the color will ALWAYS be sent, even if the person you're sending it to has color off. This can very bad – some people who do not have ANSI-compatible terminals will see garbage characters instead of colors. If the terminal correctly ignores ANSI color codes, then nothing will show up on their screen at all. The K series is mainly used to print colors to a string if the player's color level will later be tested manually (for an example, see `do_gen_com` in `act.comm.c`).

The recommended series is the CC series (i.e. `CCNRM()`, `CCRED()`, etc.) The CC series macros require two arguments – a pointer to the character to whom the string is being sent, and the minimum color level the player must be set to in order to see the color. Color sent as 'sparse' (`C_SPR`) will be seen by people with color set to sparse, normal, or complete; color sent as 'normal' (`C_NRM`) will be seen only by people with color set to normal or complete; color sent as 'complete' (`C_CMP`) will be seen only by people with color set to complete.

To illustrate the above, an example is in order:

```
#include "screen.h"
/* include screen.h in all files that you use color in */

ACMD(do_showcolor)
{
  char buf[300];

  sprintf(buf, "Don't you just love %scolor%s, %scolor%s, "
                "%sCOLOR%s!\n\r",
      CCBLU(ch, C_CMP), CCNRM(ch, C_CMP),
      CCYEL(ch, C_NRM), CCNRM(ch, C_NRM),
      CCRED(ch, C_SPR), CCNRM(ch, C_SPR));
  send_to_char(buf, ch);
}
```

What does this do? For people with color set to Complete, it prints:

```
    Don't you just love color, color, COLOR!
                       (blue)  (yellow)  (red)
```

People who have color set to Normal will see:

```
    Don't you just love color, color, COLOR!
                               (yellow)  (red)
```

People who have color set to Sparse will see:

```
    Don't you just love color, color, COLOR!
                                         (red)
```

People who have color set to Off will see:

```
    Don't you just love color, color, COLOR!
                    (no color, as you'd expect)
```

There are several common pitfalls with using the CC series of color macros:

- Do not confuse `CCNRM` with `C_NRM`. `CCNRM()` is a macro to turn the color back to normal; `C_NRM` is a color level of “normal”.
- Always make sure that every pair of “on” and “off” codes are at the same color level. For example:

  ```
  WRONG:  sprintf(buf, "%sCOLOR%s\n\r", CCBLU(ch, C_NRM),
                                        CCNRM(ch, C_CMP));
  ```

  This is wrong because if someone has their color level set to Normal, the `CCBLU` code will be sent but the `CCNRM` command will not, causing all subsequent output to be blue.

```
WRONG:  sprintf(buf, "%sCOLOR%s\n\r", CCBLU(ch, C_CMP),
                                      CCNRM(ch, C_NRM));
```

The above statement is also wrong, although not as bad. In this case, someone with color set to Normal will (correctly) not get the `CCBLU` code, but will then unnecessarily get the `CCNRM` code. Never send a color code if you don't have to – the codes are several bytes long, and cause a noticeable pause at 2400 baud.

- This should go without saying, but don't ever send color at the `C_OFF` level.
- Special precautions must be taken when sending a colored string to a large group of people – you can't use the color level of "ch" (the person sending the string) – each person receiving the string must get a string appropriately colored for his/her level. In such cases, it is usually best to set up two strings (one colored and one not), and test each player's color level individually (see `do_gen_com` in `act.comm.c` for an example).