DisplayLink Coding Task: Colour Chart

Introduction

This is a C++ coding test for potential DisplayLink programmers. It is expected that this task should take a few hours to complete.

Task description

Overview

When designing graphics hardware it is useful to be able to display various test patterns and colour ramps. The task is to write a command-line application to generate some of these test patterns. In the simplest case this program will produce a colour ramp starting with one colour on one side of the display and changing smoothly to a second colour on the other side. In the most complex case there will be a different colour in each corner of the display and each pixel on the display will show the appropriate mix of these four colours.

Input

The program should be invoked by the following command line:

```
C:>ramp.exe display tl tr [bl] [br]
```

where

- `display` is the name of the display device
- `tl` is the top left colour value
- `tr` is the top right colour value
- `bl` is the bottom left colour value [optional, defaults to `tl`]
- `br` is the bottom right colour value [optional, defaults to `tr`]

The colour values are specified as 16-bit `RGB565` pixels in hex or decimal. The bits for each colour are assigned as follows:

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Colour	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0

For example,

- White is 0xffff
- Black is 0x0000
- Pure blue is 0x001f
- Pure green is 0x07e0
- Pure red is 0xf800

The following inputs to the program are all valid:

- ramp.exe display 0x0 0x2
- ramp.exe display 65 255
- ramp.exe display 200 0 30
- ramp.exe display 0 0 3200 1800
- etc

Output

The program must fill the display with a colour ramp as defined by the input values. Each corner of the display must have the specified colour and the pixels in the middle must be a linear mix of those colours.

The image is output on the display using the `Display` class which is provided. The `get_size` method returns the size of the display and the `draw_raster` method is used to draw a row of 16-bit `RGB565` pixels on the display.

Assessment

The results of this task will be assessed in the following areas:

- Correct program operation for all input values
- Clarity of design
- Consistent coding style and demonstrated use of C++ features
- Testing strategy (Supplied unit tests, etc)
- Appropriate error handling

You must provide full source code and a brief description. The program will be tested with a variety of input values. The implementation of the `Display` class is provided below.

Notes

Correct behaviour is much more important than good performance.

This task is harder than it appears: there are a number of pitfalls and difficult cases to allow for. In particular, make sure that the colours are spread evenly when there are only a few different colours across the display. For example, a ramp from 0 to 0xf, over a width of 16 pixels, should attain each value exactly once.

Feel free to contact DisplayLink if anything is unclear or if you would like further guidance.

Display Class implementation

Display.h

```
class Display {
public:
      Display();
      ~Display();
      bool connect(const char *display_name);
      void get_size(int &width, int &height);
      void draw_raster(int x, int y,
              const unsigned short *pixels, int width);
};
```

Display.cpp

```
#include <cstdio>
#include <cassert>
#include <memory>
#include "display.h"

#define W 16
#define H  9
static unsigned short frame_buffer[W*H];

Display::Display()
{
  memset(frame_buffer, 0, sizeof(frame_buffer));
}

Display::~Display()
{
  unsigned short *pix = frame_buffer;
  for (int y = 0; y < H; y++) {
    for (int x = 0; x < W; x++) {
      if (x > 0) {
        printf(" ");
      }
      printf("%04X", *pix++);
    }
    printf("\n");
  }
}

bool Display::connect(const char *display_name)
{
  return true;
}

void Display::get_size(int &width, int &height)
{
  width = W;
  height = H;
}

void Display::draw_raster(int x, int y,
    const unsigned short *pixels, int width)
{
```

```
  memcpy(&frame_buffer[y*W+x], pixels, width*sizeof(unsigned short));
}
```