VECIMS 2005 – IEEE International Conference on
Virtual Environments, Human-Computer Interfaces, and Measurement Systems
Giardini Naxos, Italy, 18-20 July 2005

Incorporating Groups into A Mathematical Model of Color Harmony for Generating Color Schemes for Computer Interfaces

Paul Lyons and Giovanni Moretti
Institute of Information Science and Engineering
Massey University
Private Bag 11-222
5301
Palmerston North
Manawatu
New Zealand
p.lyons|g.moretti@massey.ac.nz

Abstract *– We describe an approach to developing a mathematical model of color harmony. This will be applied in the Color Harmonizer, an automated tool for coloring computer interfaces and websites. The tool will incorporate a color harmony engine that can incorporate a variety of theories for color harmony, and in the first instance, will use the rules proposed by Munsell and adapted to use in computer displays. We describe abstract and concrete color schemes, the Chromotome (a tool developed to facilitate the selection of colors) and techniques for grouping interface elements.*

Keywords *– Color Harmony, Color Harmonizer, Munsell, Chromotome, Color Selection*

1. INTRODUCTION

The authors are undertaking a long-term investigation into a technique for colorising a variety of environments using mathematical models of color harmony. Computer interfaces are the first such environment but, in principle, the same techniques could also be applied to fields such as architecture, interior design or fabric design. In earlier publications we have reported on

- The mathematical model underlying the technique
- A proof-of-concept implementation
- A tool for modelling and exploring a three-dimensional color solid

In this paper, we summarise the basic approach and report on techniques that are being developed for the current task:

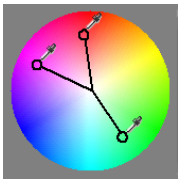

Fig. 1. A color molecule's atoms act as eyedroppers that pick colors from the color solid. This arrangement would produce a Split-Complementary color scheme.

modelling the color-relationships between members of groups of similarly colored items.

The resulting system, the Color Harmonizer, models aesthetic judgement mathematically and provides user interface developers, who generally do not have a sophisticated model of color aesthetics, with a tool that guarantees to select a harmonious set of interface colors.

2. BACKGROUND AND GENERAL DESCRIPTION

Many aesthetic and pragmatic heuristics for characterising color in computer interfaces can be mathematically formalised (following Munsell, cited in [1]) and conceptualised as an abstract color scheme which can be represented as a rigid shape, a *color molecule* that is free to move within a 3D color-space (see Fig. 1).

Locating the color molecule within the color space converts the abstract color scheme to a harmonious and pragmatically conformant concrete color scheme. The atoms in the color molecule then occupy well-defined positions in the space, thus specifying the actual colors to be used in the interface.

The Color Harmonizer operates in two phases. First, it generates the abstract color scheme by depicting essential color relationships between interface components in a rigid color molecule, with a shape corresponding to a color harmony heuristic and "interatomic distances" corresponding to the distinctions required between the interface components. Secondly, using a direct manipulation interface (the Chromotome, described below) the user locates the molecule in color space, thus determining the actual interface colors. (Fig. 1 shows a simplified representation of this interface.) The chosen colors are automatically harmonious and ensure visual distinction between components, because the standard heuristics for color harmony, as they apply to the specific target interface, have been integrated into the shape of the color molecule.

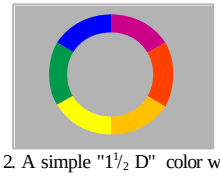

Fig 2. A simple "1½ D" color wheel

The Color Harmonizer optimizes color relationsips within 3D perceptually uniform spaces [2], because these can capture more sophisticated color relationships than conventional color wheels (Fig 2), which represent few colors, and are imprecise about the relationships between them. Distances in perceptually uniform color spaces are proportional to perceived differences between color-pairs. Commonly used color spaces, particularly RGB, are perceptually non-uniform, so constant color relationships are not easily achievable within them. However, RGB coordinates can be converted to coordinates in the L*u*v* space (Fig. 3) [3, 4] which is usually accepted as being approximately perceptually uniformity [5].

Munsell developed Runge's [1] and Goethe's [5] 3D color solids into a color classification system based on an empirical arrangement of colors in a 3D space. His numeric color-"naming" system was adopted by many color classification bodies, leading to perceptually uniform color spaces such as the CIE's L*u*v* space.

The first dimension of this solid is *hue*, which specifies the basic color-name for an image component – whether it is red, orange, yellow, green, blue, or purple. The second color dimension is *value* – the amount of light an image reflects or emits. Finally, *chroma* or *saturation*, distinguishes dull (brown) from vivid (orange). If all three dots in a pixel are turned on equally, its saturation is 0; it is grey. If one dot is gradually turned off, the pixel's hue will depend on the relative values of the other two dots, and it will gradually increase to full saturation.

Distances in any 3D color space will provide some color difference metric. Our work, however, depends on an ability to manipulate color differences in the abstract. The mapping between distances in the color space and perceived color differences should therefore be consistent.

Color differences are measured in terms of multiples of the JND, the Just Noticeable Difference. This basic unit of color difference has been psychometrically determined in all regions of color space [2]. The perceived color difference between colors at the end-points of a line of a given length in a perceptually uniform color space should correspond to an equal number of JNDs, irrespective of the location of the line within the space. Most color spaces sacrifice this important internal property for a pleasing external symmetry (spherical, cylindrical, etc.). However, Munsell's experiments showed that a perceptually uniform color space is non-symmetric. Farnsworth [6] has improved on Munsell's original space. We work with the L*u*v* space which combines approximate perceptual uniformity and mathematical tractability.

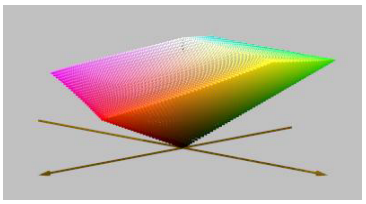

Fig. 3. The L*u*v* color space

Munsell developed a color harmonisation algorithm that exploits the geometry and perceptual uniformity of his 3D color solid. Like Itten [7], he theorised that harmonious images will "centre" on (middle-value) grey. Complementary colors *a* and *b* will balance if $cs_a . \sum area_a = cs_b . \sum area_b$, where cs_a (color strength of *a*) is $saturation_a \times lightness_a$ (saturation and lightness range from 1 to 10). Thus a small area of *a* with high color strength will balance a large area of *b* of low color strength.

This approach was impractical when first proposed – it required practitioners to measure the hue, saturation and value of each color in an image; and to calculate the exact areas occupied by each color. At the time the technique was proposed - 1905 - the images in question were painted or printed, and there were no computers to take over the drudgery, so the technique was not widely used (although it did enjoy some popularity with printers whose "images" – a page of text with a few wingdings or dropped capitals - were relatively simple). It also coincided with the Fauve movement in art. Fauve is a French word for "wild animal," and referred to an artistic style that replaced subdued or naturalistic coloring with highly saturated, unrealistic colors designed to shock the viewer. Today, such images do not have the same shock value, but at the time they seemed challenging, even offensive. In such an artistic climate, Munsell's approach, with its beautifully harmonised, delicate colors, was doomed to obscurity.

Now that computers can generate images, neither the measurement of color parameters nor the subsequent calculations are problematic. There are, however, pragmatic considerations that Munsell – whose focus was on fine art painting – did not take into account. He did not allow for the fact that text has to be legible against its background, nor that in a computer interface, some visual components have to be clearly distinguishable from others, or should be linked by their color into a (subconsciously) identifiable group, nor that colors of some components in one window need to be identical in another interface window, although the windows otherwise contain quite different images. For example, *OK*, *Close* and *Cancel* buttons should be consistently colored across an application, and should also harmonize chromatically with the images in the individual windows.

The Color Harmonizer approach takes these pragmatic considerations and Munsell's aesthetic rules into account.

The current project focuses on the development of a system that will allow a designer to harmonize the colors of an interface in two phases. In the first phase, an abstract color scheme is produced. This can be visualised as a rigid "molecule" that is free to be rotated and positioned anywhere in the 3D color space. Each interface component is represented as an "atom" on the molecule, and the position of an atom in the 3D color space determines the color of the corresponding interface component. Because of the rigidity of the molecule, the spacing, and thus the color relationships between its "atoms" are constant.

In the second phase, the designer converts the abstract color scheme into a concrete color scheme by positioning and orienting the color molecule in the 3D space. All the interface colors change in real time as this is happening.

The rigid molecule incorporates all the pragmatic and aesthetic considerations about the interface color scheme, so it is also possible to allow the eventual user of the software, as well as its designer, to reposition and reorient it in the color space while maintaining a pleasing set of colors, and without losing the clarity of the interface components.

3. PREVIOUS WORK

When complete, the system described above will be capable of capturing the characteristics of a complex interface, generating an abstract color scheme with a spring-based optimisation algorithm [8], and applying the resulting abstract color scheme in real time to an interface.

Preliminary implementations of some parts have been completed. In particular, a static demonstration [9], using an image of a circuit diagram, has been produced. It allows the diagram to be recolored in real time, ensuring that text is always visible on its background, and that important items such as connection points and wires are always clearly distinguishable.

The color combinations produced often involve subtle colors that are difficult to name ("colored grays", in painters' parlance). They are usually harmonious, but occasionally not. In such cases, the strength of the technique is that it takes only a moment, and a single interface operation, to move the color molecule to another position and orientation, corresponding to another concrete color scheme that will probably be an improvement. Conventional color choosers such as the Windows® color picker allow the user to alter one color at a time, and can involve hours or even days of fine tuning to produce a complete color scheme. More sophisticated pickers such as those surveyed in [10] exist, but none permit the designer to characterise an interface as the Color Harmonizer does.

The introduction to this paper mentions three areas of completed work, which we will now describe briefly.

- The mathematical model underlying the technique
- A proof-of-concept implementation
- A tool for modelling and exploring a 3D color solid.

3.1 The Mathematical Model

The system has three major mathematical modules, the harmony model proper, the allocater, and the optimizer.

The harmony model estimates the color harmony of a given image. As formulated, Munsell's rules apply to images involving only two colors, and determine the distance between their area-weighted "center of gravity" and mid-gray. Our version simply substitutes a calculation of the centre of gravity of a pixel-based image containing an arbitrary number of colors.

This has been implemented in our proof-of-concept implementation (below). Color schemes produced using this software are usually harmonious, and occasionally garish. The garish schemes seem to involve combinations of light versions of "naturally" dark colors such as red and blue, with dark versions of "naturally" light colors such as yellow. This suggests that Munsell's scheme should be modified to weight color schemes in which colors are closer to their natural values more highly. Others have suggested that Munsell's rules overemphasize saturation and value. Consequently, in the actual implementation, the harmony model is a replaceable module so that we can incorporate other rulesets besides Munsell's.

The optimizer works in collaboration with the harmony model. It alters the location of the colors of interface components within the color space to maximize the color harmony of the whole interface.

The allocater (not currently complete) will determine the initial positions of the color "atoms" that represent the interface components on a wireframe "molecule." Since the atoms cannot slide past each other along the wires of the molecule, it is important that they are initially positioned so that interface items that need to be distinct are not adjacent. The allocater captures three parameters about each interface component; its area, its importance, and its longevity, and allocates it a color strength based on these parameters. The color strength of a component is directly related to its importance and inversely related to its area and longevity.

The allocater positions atoms for low colour-strength components close to the centre of the 3-D colour solid, and atoms for high colour-strength components closer to its extremities.

There is some choice about the order in which the atoms are added, so the allocater produces a number of alternative initial configurations, and the optimizer optimizes each of these independently. Subsequently, the user is free to alter the position and orientation of each of these configurations to produce separate concrete color schemes.

3.2 The Proto Harmonizer

The Proto Harmonizer is a proof-of-concept implementation that demonstrates the validity of the approach using a static image [9]. The image is a circuit diagram in which electrical modules (colored rectangles with a name and a

number of small triangular terminals) are connected to other modules by lines representing wires. The image has the following interface components: background, electrical modules, module names, terminals, and wires. The electrical modules are rectangular with textual names and terminals on their edges. Wires interconnect the terminals. The implementation is restricted in several ways:

- The circuit cannot be edited.
- The order of atoms on the wireframe is fixed.
- The list of interface components and their properties are hardcoded and not captured dynamically.
- It only produces complementary color schemes.

However, the colors are recalculated in real time as the user repositions the wireframe within the space. This gives a real feeling of being in control of the color scheme, and makes it easy for the user to experiment with colors. The experimentation is not always successful, but because the interface can be recolored almost instantaneously with a high probability of producing harmonious color, the occasional garish result is only momentarily disconcerting.

3.3 The Chromotome – A Tool for Modelling the Color Solid

The success of this work rests on several foundations. We have covered realtime interface updating and a color choice algorithm that takes account of interface characteristics. Another important support is an intuitive tool for exploring 3D color space, to position and orient the abstract color scheme. We have developed the Chromotome [11], an interface component that shows a color space, and allows a user to display and re-orient an abstract color scheme within that space. The Chromotome is thus the tool which the designers and users of an application can use for converting the abstract color scheme into a concrete color scheme and for altering a color scheme should the current color scheme grow tiresome.

We have previously explained the benefits of working in a non-symmetric perceptually uniform color space such as the L*u*v* color space. Why then does the Chromotome (Fig. 4) present a spherical color solid to the user?

Because perceptually uniform spaces have strange shapes, and when this is further modified by a cutaway, the continually varying profile that is displayed as the shape is rotated looks so weird that it disturbs the user's ability to concentrate on the color selection task. Consequently the abstract color scheme is optimized in a perceptually uniform coordinate system and mapped onto the Chromotome's spherical color space for display.

The downside to this is that the color molecule changes shape as the Chromotome rotates. However, this is expected to be less distracting than the varying profile described above.

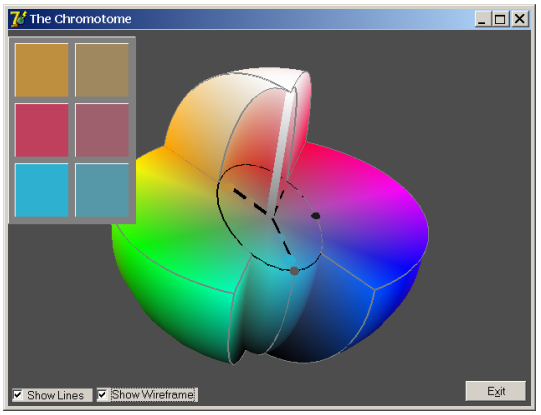

Fig. 4. The Chromotome allows a color molecule to be positioned in a cutaway view of a spherical color solid.

3.4 Groups of Screen Elements

Some screen elements naturally form groups. Consider the simple web interface shown in Fig. 5. Its layout style, with a left-hand navigation bar, is widely used for websites, and a similar model has begun to appear in desktop software; for example recent versions of Microsoft Office™ can display a Taskpane down the side of the window. In most interfaces, the buttons in the navigation bar will have identically colored text, and the color of each button's background would also be the same.

It might seem simple to group these items, so that they can be treated identically, but the Color Harmoniser's optimisation relies on a set of properties associated with each screen item, and it is necessary to allocate these properties carefully in the case of groups of items.

There is another reason for grouping items – minimising the effort involved in data capture. Even in the simple web page shown in Fig. 5 there are seven basic items each with its own color (header text, header background, footer text, footer background, navigation bar background, body text and body background).

If the buttons are treated individually, there are another ten - 5 x ($button_i$ text, and $button_i$ background) giving a total of 17 items each with size, longevity and importance parameters, and the interactions between one element and all the others. The tedium of defining the interaction and properties of an interface is greatly reduced if it is possible to treat all the buttons as a single group item ("Navigation Buttons"). This concept could be extended into hierarchies of screen elements, although at this stage it is not clear whether there would be benefits from this or not.

From the composite properties of a group, the Color Strength of each group is calculated and used during an optimization phase to move the points representing the colors of the groups and primitive screen items to positions in the color space that satisfy the constraints associated with

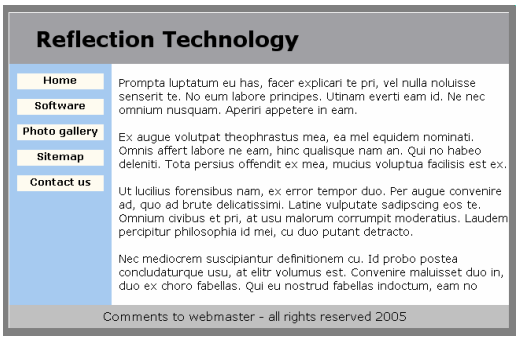

Fig. 5. An interface with grouped items

a particular type of color scheme. This ensures that those screen items or groups that should be visually distinct really are distinct.

3.5 Groups – the implications for optimization

The designer allocates an interaction to each pair of primitive screen elements: *don't care* (if they are semantically unrelated, and physically separated, so that their colors only need to have an aesthetic relationship), *same* (if they are semantically related so that their colors should be identical) and *distinct* (if they overlap and need to be clearly distinguishable from each other; text on a background is the prime example of this).

For atoms in a group, the *don't care* interaction isn't relevant – the items have been grouped because they are to be *colored as a group*. The other two types of interaction – *same* and *distinct* - have a subtly different interpretation within a group of atoms.

3.6 Groups with identically colored elements

The meaning of *same* is obvious – all the elements in the group will be colored identically Fig. 6 shows a molecule with a pseudo-atom representing the navigation buttons.

During the optimization phase, when determining the spacing of the items within the molecule, a pseudo-atom representing a *same colored group* is treated as a single interface element usually with the area of a single member of the group (though other strategies exist). After the optimization, when the molecule is positioned and oriented within the color space using the Chromotome, the color associated with the position of the pseudo-atom will be applied to all members.

A beneficial side effect of using pseudo-atoms to represent groups of screen objects is that the optimization involves fewer items, and thus takes less time

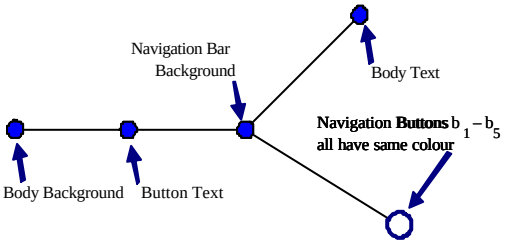

Fig. 6. Agroup of identically colored screen elements

3.7 Groups with differently colored elements

The other case involves groups whose element interactions have been classified as *distinct.* In the simpler case we have been discussing above, a group with a *same* interaction is chromatically harmonised with other atoms and pseudo-atoms in the molecule representing the color scheme for the whole interface. In this more complex and interesting case, the color scheme for the atoms in the group needs to be internally harmonious. It is therefore appropriate to generate an abstract color scheme for the group using a subsidiary color molecule.

This alternative is illustrated in Fig 7, where each atom (c1-c5) is arranged in the color space about some central point (C). This type of situation would occur when a subset of the items in an interface were to be visually related by being given their own color scheme – a sub-scheme.

There are various ways the group could be colored. Most simply, it could be optimized as though all the atoms really were at C, and then when the coordinates of C had been determined, the subgroup could be arranged about this centre of gravity. In effect this would treat the group as a totally local phenomenon, simply pushing the atoms (c1-c5) away from the central point C by some measure deemed to give sufficient visual difference. For example, in Fig 7 the group is arranged in a circular pattern around point C, giving a form of analogous color scheme (all of the points c1-c5 will be similar to but distinct from each other). This would be simple to implement but the local distribution of atoms around C to c1-c5 does not take into account any atoms other than C.

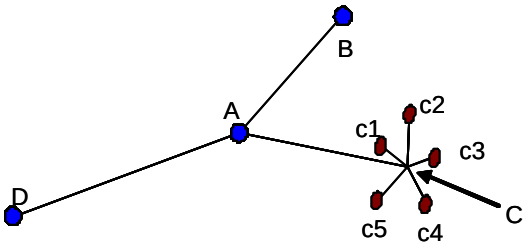

Fig 7. A split complementary molecule with an independently colored subgroup

If the colors are to be visually distinct, then the color differences within the group must have a lower limit that relates to the size of the objects represented by c1-c5 (it's easier to see subtle color differences between large areas than between small areas). While there's a lower bound on the separation of the members of the set c1-c5, there is also an upper bound - if it is too large, then atoms on the group may come too close to A. Depending on the plane upon which c1-c5 are organised (if indeed it is a plane) then a member of the group could lie directly on the A-C axis and so could eventually become visually indistinguishable from point A.

The situation is made more complex in that point A itself could be an individually colored group whose atoms are expanding away in the color space from point A. There are upper and lower limits on the size of subgroup wire frames, both imposed by the need to keep items visually distinct. Another aesthetic consideration is that if the atoms c1-c5 are arranged in a subgroup, then the designer probably intends these items to be, in some sense, related. Keeping them close together in the color space will, in the viewer's mind, cause them to be associated with each other because of the similar coloring. If the group expands too much, this "association by proximity" will be lost.

The designer may wish to specify the orientation of a group's submolecule, or this could be controlled by the optimisation process. For example, the designer may specify that c1-c5 lie on a plane that is normal to the axis A-C.

The case outlined above suggests that groups should be defined with respect to a central point (e.g C in Fig 7) but optimized globally. To accomplish this, a form of spring optimization described by Ryall et al [12] will be used to optimise the overall arrangement while ensuring that groups remain cohesive by using springs to anchor group members to a the centre of a group (point C in Fig 7).

4. CONCLUSION

Introducing groups of colored elements into an automatic coloring algorithm introduces significant benefits for the user both in terms of simplicity – it's easier to describe complex interfaces, and allows the possibility of visually distinct but harmoniously colored subgroups related to the primary color scheme. The price of this is the need to globally optimize all the interface elements, using the groups to set the initial position and optimization constraints but optimizing over the complete set. Spring-based optimization algorithms have the flexibility to cater for the local and global nature of the constraints.

REFERENCES

[1] F. Birren, *MUNSELL: A Grammar of Color*: Van Nostrand-Reinhold, 1969.

[2] D. L. MacAdam, "Visual sensitivities to color differences in daylight," *J. Opt. Soc. Am.*, vol. 32, pp. 247-273, 1942.

[3] G. J. Chamberlin and C. D.G., *Colour - Its Measurement, Computation and Application*: 1980, 1980.

[4] D. Travis, *Effective Color Displays: Theory and Practice*: Academic Press 1991, 1991.

[5] R. B. Norman, *Electronic Color*: Van Nostrand Reinhold, 1990.

[6] Farnsworth, "A temporal factor in colour discrimination; Visual Problems of Colour, II (cited in Wyszecki and Stiles, 1967)," presented at Nat. Phys. Lab. Symp. No. 8, 1958.

[7] J. Itten, *The Art of Colour; New York*: New York: Van Nostrand Reinhold, 1973.

[8] K. Ryall and J. Marks, "An Interactive System for Drawing Graphs," presented at 1996 Graph Drawing Symposium, 1996.

[9] G. Moretti, P. Lyons, and M. Wilson, "Chromatic Interpolation for Interface Design," presented at OZCHI 2000, Sydney, Australia, 2000.

[10] P. Lyons and G. Moretti, "Nine Tools for Generating Harmonious Colour Schemes," presented at 6th Asia Pacific Conference on Computer Human Interaction (APCHI 2004), Rotorua, New Zealand, 2004.

[11] G. Moretti, P. Lyons, and M. Wilson, "Chromotome: A 3D Interface for Exploring Colour Space," presented at 6th Asia Pacific Conference on Computer Human Interaction (APCHI 2004), Rotorua, New Zealand, 2004.

[12] K. Ryall, J. Marks, and S. Shieber, "An interactive constraint-based system for drawing graphs " in *Proceedings of the 10th annual ACM symposium on User interface software and technology* Banff, Alberta, Canada ACM Press, 1997 pp. 97-104