Improved HCI Color Model Specifications

1. INTRODUCTION

Given an RGB color space, there exist various representations in cylindrical coordinates, most notably hue–saturation–lightness/brightness (HSL/HSB) model and hue–saturation–value (HSV) model.

These models once received massive uses for their simplicity in early ages of computer graphics and is still common among amateurs of graphic design, photograph manipulation, etc. However, by looking into these models, they clearly show severe defects. Thus, users who have certain quality requirements have renounced them, and more advanced color spaces are adopted instead.

One of the good examples is the hue–chroma–luminance (HCL) model, which features a direct transliteration from CIE LAB color space using the intuitive parameters. Nonetheless, its complex nature with human-unpredictable behaviors regarding conversion to an RGB space poses some difficulties on implementation.

Considering those problems, we call for an improved cylindrical representation for an RGB space. The solution is discussed below.

2. REVIEW TO HSL MODEL

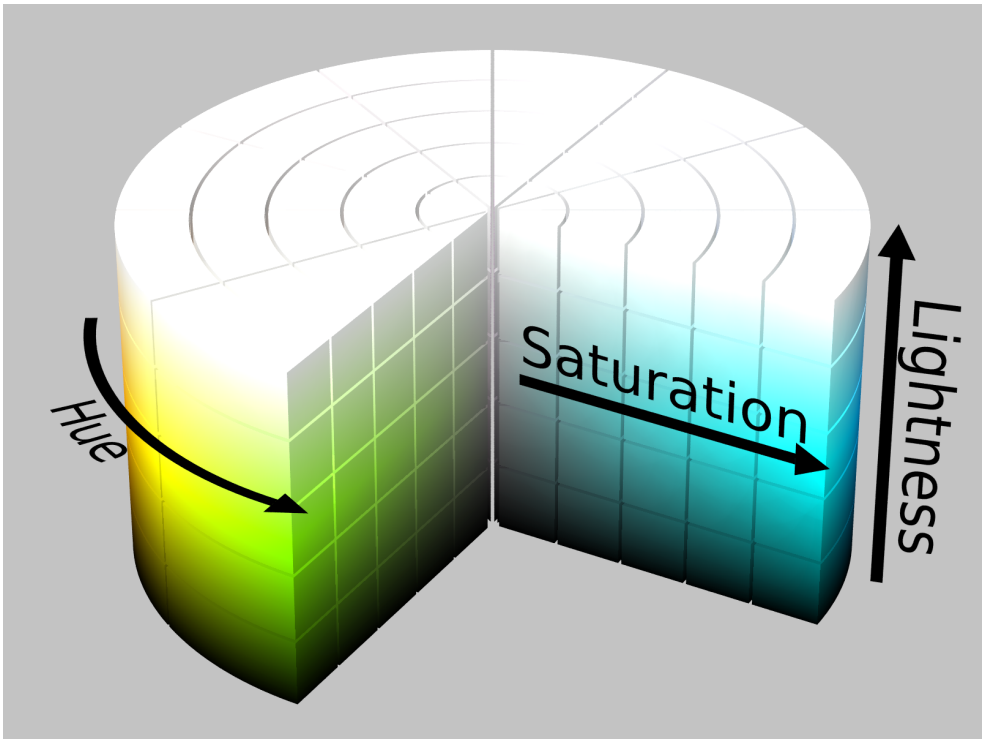

Figure 1: HSL cylinder

It is necessary to review HSL since our solution is based on it. See Fig. 1.

2.1. Qualitative Description

HSL has a cylindrical geometry, with hue, the angular dimension, starting at the red primary at 0°, passing through green at 120° and blue at 240°, and then wrapping back to red at 360°. The central vertical axis is called the neutral axis as it comprises the neutral (or achromatic, gray) colors, ranging from black at lightness 0, the bottom, to white at lightness 1, the top.

All the primary and secondary colors and their linear mixtures between adjacent pairs are arranged around the outside edge of the cylinder with saturation 1. These saturated colors have lightness 0.5, and mixing them with black or white leaves saturation unchanged.

Because these definitions of saturation – in which very dark or very light near-neutral colors are considered fully saturated – conflict with the intuitive notion of color purity, often a biconic solid is drawn instead, as shown in Fig. 2.

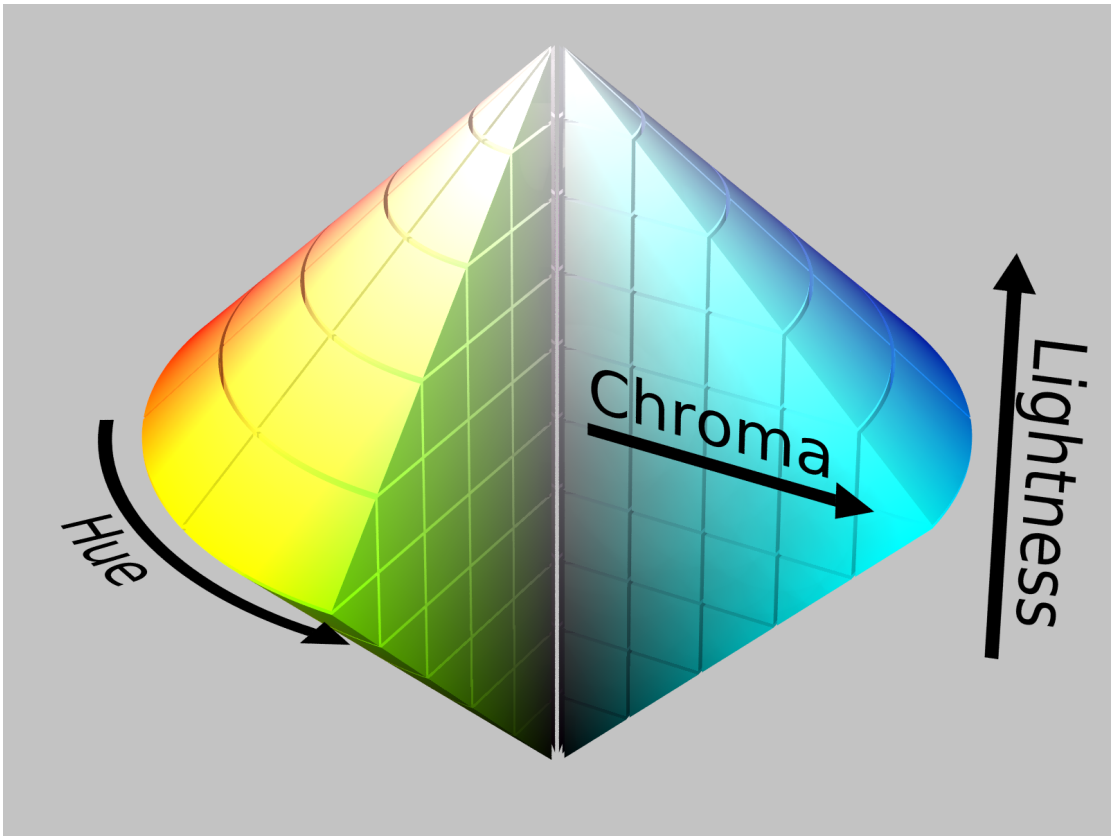

Figure 2: HSL Cone

Here the term *chroma* is used for distinction, though *saturation* is still commonly referred to in this pattern. As an intermediate variable, calculating chroma can be useful in developing such models.

2.2. Specifications

Specifically, the approach to transform RGB to HSL and the definition of the three parameters hue H, saturation S, and lightness L are shown as follows.

Based on the fact that any RGB color space can be seen as a perfect unit cube in a Cartesian coordinate system, where r, g and b are the three axes, the cube is firstly rotated about the origin so that the neutral axis is on one of the coordinate axes – the vertical axis assumed by convention.

Next, the tilted cube is projected from above onto a horizontal plane called the chromatic plane. Every point on this plane is considered to have equal brightness, rendering a sole representation of color, hence the name. Working with this plane yields the benefit of much easier work, yet the brightness difference between primaries and secondaries is lost through the projection.

Using this approach, the maximum M, minimum m and chroma C are calculated in advance.

$$\begin{aligned} M &= \max(r, g, b) \\ m &= \min(r, g, b) \\ C &= M - m \end{aligned}$$

Then, the lightness is defined as

$$L = \frac{1}{2}(M + m)$$

The hue has an awkward piecewise definition as

$$H = \begin{cases} \text{undefined}, & \text{if } C = 0 \\ 60^\circ \dfrac{g-b}{C} \bmod 360^\circ, & \text{if } C \neq 0, \quad M = r \\ 60^\circ \dfrac{b-r}{C} + 120^\circ, & \text{if } C \neq 0, \quad M = g \\ 60^\circ \dfrac{r-g}{C} + 240^\circ, & \text{if } C \neq 0, \quad M = b \end{cases}$$

In geometric interpretation, the method involves warping the hexagon chromatic plane into a circle, as in Figure 3.

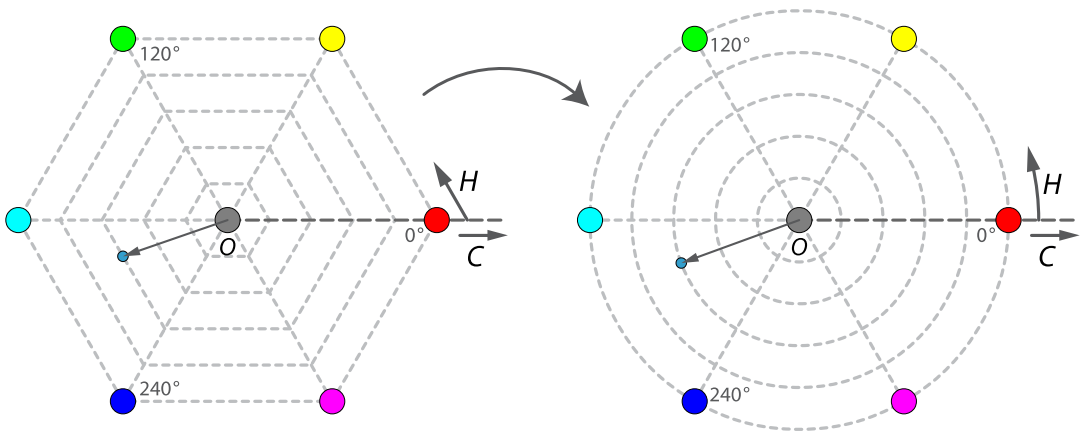

Figure 3: Warping Hexagon into Circle

The saturation is also defined piecewise, as

$$S = \begin{cases} 0, & \text{if } L = 0 \vee L = 1 \\ \dfrac{C}{1 - |2L - 1|}, & \text{otherwise} \end{cases}$$

This indicates a scaling of chroma to fill the interval [0, 1] for every combination of hue and lightness, as in Fig. 4.

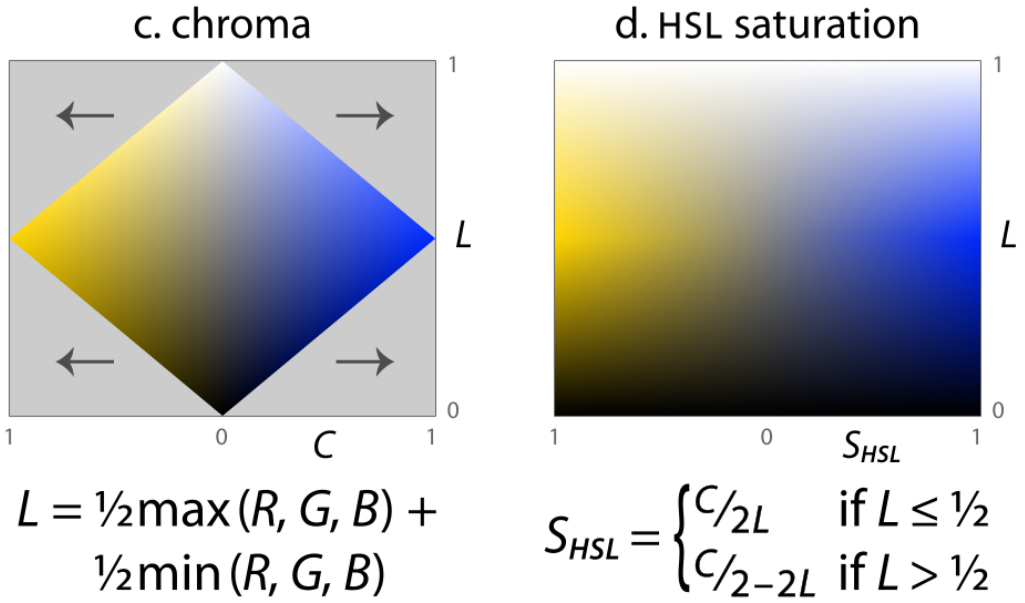

Figure 4: Scaling to Fill the Interval

It is clear that the scaling scheme affects the least on entries near the middle, while extremely dark or bright entries undergo a *hyperbolic* deformation.

It can be concluded that every definition of the parameter in HSL involves some extent of distortion. While the warp of hue is relatively small, the projection renders a remarkable error in the lightness for primaries and secondaries, and worst of all, the scaling to chroma for saturation proves to be fatal.

3. DESIGN OF IMPROVED HCI MODEL

We propose the model called Improved Hue–Chroma–Intensity (HCI) to mend the problems in the current HSL model. Although our results may have been re-produced by some existing models, the definitions are different. The design is shown as follows.

3.1. Design Principles

(1) Define a measurement for brightness to effectively discriminate primaries and secondaries.

(2) Perceptual brightness (or *luminance*) should not be taken into consideration, mainly because a general RGB model does not give what red, green and blue actually are, as well as their relationships.

(3) Drop saturation, adopt chroma so that the number reflects colorfulness more objectively. Off-base i.e. darker and brighter colors should not be measured as fully colored.
(4) Redefine hue in an intuitive way and try to avoid any warping and piecewiseness.
(5) Use simple, understandable definitions to grant high accessibility to most people.

3.2. Tilted Cube Model

We take the same tilted cube as in HSL model, with the concept of neutral axis directly inherited. However, since distortion, including forcing primaries and secondaries into one plane, warping hexagons into circles, and non-linear scaling, is not tolerated, we simply contain the cube into our cylinder.

The cylinder is determined by the range of our three parameters: hue (H), chroma (C), and intensity (I). Saturation is dropped as discussed above.

We will first define these parameters based on this model. Next, we will give the conversion formula for any given point entry (r, g, b) in the RGB space to (H, C, I) in the Improved HCI space, where

$$0 \leq r, g, b \leq 1$$

3.3. Intensity

Intensity is one of the parameters we use in our model, denoted I. The parameter takes a simple definition – the arithmetic mean of the three components.

$$I = \frac{1}{3}(r+g+b)$$

This means we project any point in the tilted cube onto the neutral axis and take the vertical height as its intensity, compressed linearly into the range

$$0 \leq I \leq 1$$

Since the projection preserves the information of relative vertical height, it avoids the problem that secondaries measure the same brightness as corresponding primaries. For example, pure red (1, 0, 0) has $I = 1/3$, and pure yellow (1, 1, 0) has $I = 2/3$. Actually, pure orange (1, 0.5, 0) has $I = 0.5$, which remains the same as in HSL.

3.4. Chromatic Planes

The chromatic plane in our model refers to the plane on which every entry has the same intensity I. By this definition, if we clip the tilted cube with a horizontal plane, the intersection is a chromatic plane. All these planes have vertical normals, most notably the neutral axis.

In general, there is a one-to-one correspondence between a value of intensity and a chromatic plane. A chromatic plane is either a regular triangle or a hexagon, depending on the intensity chosen. It is a hexagon if $1/3 < I < 2/3$, otherwise a regular triangle. The special cases are $I = 0$ and $I = 1$, where the triangle degenerates into a point.

Since the center of every chromatic plane is on the neutral axis, we call it the neutral point. We also name the chromatic plane corresponding to $I = 0.5$ as the base plane, which is the only regular hexagon among the set.

These concepts are important so as to define chroma and hue, as shown below.

3.5. Chroma

Before defining chroma, we introduce *deviation*, denoted D, which is the distance between the entry point and the neutral axis. Since D has the maximum $\sqrt{6}/3$, we stretch it linearly to fill the interval from 0 to 1 and define it as the chroma C. Thus,

$$\begin{aligned} D &= \sqrt{(r-I)^2+(g-I)^2+(b-I)^2} \\ &= \sqrt{\frac{1}{3}[(r-g)^2+(g-b)^2+(b-r)^2]} \\ C &= \frac{\sqrt{6}}{2}D \end{aligned}$$

Deviation of the form independent of intensity is favorable due to its better interoperatability in case the definition of intensity is substituted.

An equivalent yet more widely recognized method is to find the Cartesian coordinates $\boldsymbol{C} = (C_x, C_y)$ of the entry on the chromatic plane and then the distance to the neutral point. Thus,

$$\begin{aligned} C_x &= r-\frac{1}{2}g-\frac{1}{2}b \\ C_y &= \frac{\sqrt{3}}{2}(g-b) \\ C &= \sqrt{C_x^2+C_y^2} \end{aligned}$$

This method does not require stretching because C_x and C_y has been normalized with the coefficient C/D.

Either method yields the result

$$C = \sqrt{\frac{1}{2}[(r-g)^2+(g-b)^2+(b-r)^2]}$$

By this definition, the chroma reaches 1 only if the entry is pure primaries or secondaries. A pure tertiary has the maximum chroma $\sqrt{3}/2$ on the base plane.

Because chromatic planes are not circular, an entry on a given chromatic plane risks to exceed the limit of chroma depending on the specific color. An arbitrary entry has the maximum of chroma C_m safe to be assigned. Given a chromatic plane, C_m is equal to the radius of its inscribed circle scaled by the coefficient C/D. It satisfies the following equation.

$$C_m = \frac{3}{4}(1-|2I-1|)$$

Saturation, if needed, may be defined as C/C_m, ranging from 0 to 4/3.

3.6. Hue

Lastly, we define the hue H if the entry is not on the neutral axis. On the chromatic plane, we have vector $\boldsymbol{c}_i = (1, 0)$ from the neutral point to pure red and $\boldsymbol{C} = (C_x, C_y)$ from

the neutral point to the entry. The hue H is defined as the angle from $\boldsymbol{c}_i$ to $\boldsymbol{C}$. One can easily write

$$\cos H = \frac{C_x}{C}$$

Solving for H, we obtain

$$H = \begin{cases} \arccos\dfrac{2r-g-b}{2C}, & \text{if } g \geq b \\ 360^\circ - \arccos\dfrac{2r-g-b}{2C}, & \text{if } g < b \end{cases}$$

There exist several alternative expressions with their respective complexities. For programming, if the two-argument arctangent function (atan2 (y, x), written arctan2 below) is available, it may be expressed as

$$H = \arctan 2\left(\sqrt{3}(b-g), -(2r-g-b)\right) + 180^\circ$$

3.7. Summary and Backward Conversion

In summary, the variables in Improved HCI model are hue H, chroma C, and intensity I. The formula for forward conversion is:

$$C = \sqrt{\frac{1}{2}\left[(r-g)^2 + (g-b)^2 + (b-r)^2\right]}$$

$$H = \begin{cases} \text{undefined}, & \text{if } C = 0 \\ \arccos\dfrac{2r-g-b}{2C}, & \text{if } C \neq 0, g \geq b \\ 360^\circ - \arccos\dfrac{2r-g-b}{2C}, & \text{if } C \neq 0, g < b \end{cases}$$

$$I = \frac{1}{3}(r+g+b)$$

To find the backward conversion, we solve the equations above with respect to r, g, and b, obtaining

$$r = I + \frac{2}{3} C \cos H$$

$$g = I + \frac{2}{3} C \cos(H - 120^\circ)$$

$$b = I + \frac{2}{3} C \cos(H + 120^\circ)$$

if the hue is defined.

If the hue is undefined, the chroma is 0. Entering an arbitary value of h, the formula still holds, and the result is

$$r = g = b = I$$

4. GENERALIZATION

Since the model is irrelevant to the specification of the primaries, it can apply to any color space defined by three primaries. That is, given any three primaries, the formulae still work in that space.

5. APPLICATIONS

Color palette

Improved HCI is designed specifically for the standard RGB palette for LibreOffice.

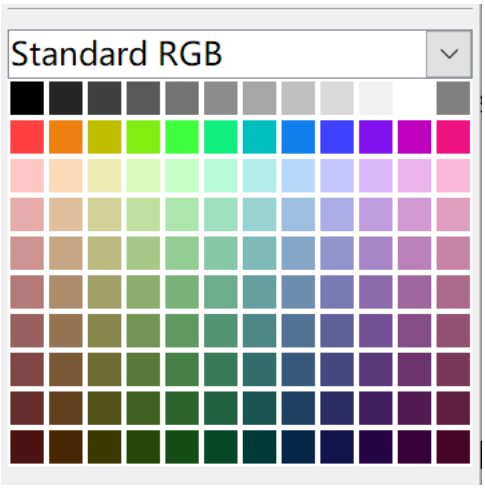

Figure 5: Palette Created with Improved HCI

The first row contains a grayscale with intensity $I = 0.00, 0.15, 0.25, \ldots, 0.85, 0.95, 1.00$, plus 0.50.

The second row contains highly saturated colors with hue $H = 30°n$, n being a natural number, chroma $C = 0.75$, and intensity $I = 0.50$.

The rows three to ten contains moderately saturated colors with hue H the same as above, chroma $C = 0.225$, intensity $I = 0.85, 0.75, \ldots, 0.25, 0.15$.

Developing other color models

Improved HCI plays a significant role in the development of Balanced RYB color model.

APPENDIX 1. CREDITS

Key Authors
Flora Canou
Alexander Zheng

Reference Materials
Constructing Cylindrical Coordinate Colour Spaces, by Allan Hanbury, 2008

This article uses material from the Wikipedia article *HSL and HSV*, which is released under the Creative Commons Attribution-ShareAlike 3.0 Unported License.

Figure 1 by SharkD; CC BY-SA 3.0.
Source
Figure 2 by Jacob Rus, SharkD; CC BY-SA 3.0.
Source
Figure 3 by Jacob Rus; CC BY-SA 3.0.
Source
Figure 4 by Jacob Rus; CC BY-SA 3.0; clipped.
Source
Figure 5 by Flora Canou; CC BY-SA 3.0.

APPENDIX 2. EXAMPLE CONVERSION PROGRAM IN C

Note: these codes are licensed under the Mozilla Public License 2.0.
It is recommended to copy them to a text editor to read these codes.

```
/* Copyright 2018-2019 Flora Canou, Alexander Zheng | V. C5-1.3.0
| Improved HCI to RGB Convertor
 * This Source Code Form is licensed under the Mozilla Public Li-
cense, v. 2.0.
 * If you have not received a copy of the license, visit https://
mozilla.org/MPL/2.0/.
 * This program converts entries between Improved HCI and RGB
color models.
 * See Improved HCI Color Model Specifications for more info.
 */

#include <stdio.h>
#include <math.h>
#define TAU   6.28318530718 //2*PI

double ihciForth_hue (double p1, double p2, double p3, double C)
{
```

```
    double H; //hue
    if (p1 == p2 && p2 == p3)
        return -1; //gray, hue is undefined
    else if (p2 >= p3)
        H = acos ((2*p1 - p2 - p3) / (2*C));
    else
        H = TAU - acos ((2*p1 - p2 - p3) / (2*C));
    return (H);
}

void ihciForth (double p1, double p2, double p3, double *H, dou-
ble *C, double *I)
{
    *C = sqrt (((p1 - p2)*(p1 - p2) + (p2 - p3)*(p2 - p3) + (p3 -
p1)*(p3 - p1)) / 2);
    *H = ihciForth_hue (p1, p2, p3, *C);
    *I = (p1 + p2 + p3) / 3;
}

void ihciBack (double H, double C, double I, double *p1, double
*p2, double *p3)
{
    *p1 = I + 2*C * cos (H) / 3;
    *p2 = I + 2*C * cos (H - TAU / 3) / 3;
    *p3 = I + 2*C * cos (H + TAU / 3) / 3;
}

void instruct (void);
int main (void)
{
    instruct();
    int mode = 0;
    double order[3];
    double H, C, I; //hue in degrees, intensity, chroma
    double r, g, b; //red, green, blue

    while (1)
    {
        if (mode)
            printf ("Mode: forward (RGB to IHCI). \n");
        else
            printf ("Mode: backward (IHCI to RGB). \n");
        printf ("> ");
        scanf ("%lf %lf %lf", &order[0], &order[1], &order[2]);
        if (order[0] == -1)
        {
```

```
            if (order[1] == 0 && order[2] == 0)
            {
               mode = (mode + 1) % 2;
               continue;
            }
            if (order[1] == -1 && order[2] == -1)
               return 0;
         }
         if (mode) //RGB to IHCI
         {
            r = order[0]; g = order[1]; b = order[2];
            ihciForth (r, g, b, &H, &C, &I);
            if (H == -1)
               printf ("Hue is undefined, \nChroma\t\t= %f, \nIn-
tensity\t= %f. \n", C, I);
            else
               printf ("Hue\t\t= %f degrees, \nChroma\t\t= %f, \
nIntensity\t= %f. \n", H*360 / TAU, C, I);
         }
         else //IHCI to RGB
         {
            H = order[0]*TAU / 360; //degree to radian
            C = order[1]; I = order[2];
            ihciBack (H, C, I, &r, &g, &b);
            printf ("Red\t= %f, \nGreen\t= %f, \nBlue\t= %f. \n",
r, g, b);
         }
      }
}

void instruct (void)
{
   printf ("This program converts entries between Improved HCI
and RGB color models. \n"
         "In forward mode, enter the values for red, green and blue,
respectively. \n"
         "They should be between 0 and 1. \n"
         "In backward mode, enter the values for hue, chroma and in-
tensity, respectively. \n"
         "The hue should be in degrees, between 0 and 360. The other
two should be between 0 and 1. \n"
         "In case the hue is undefined, enter any value for hue and
0 for chroma. \n"
         "The entries should be separated with a space. \n"
         "Enter -1 0 0 to toggle modes. Enter -1 -1 -1 to exit. \n\
n");
```

```
}
```