Improved RGB Color Space Cylindrical Representation

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 imposes 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

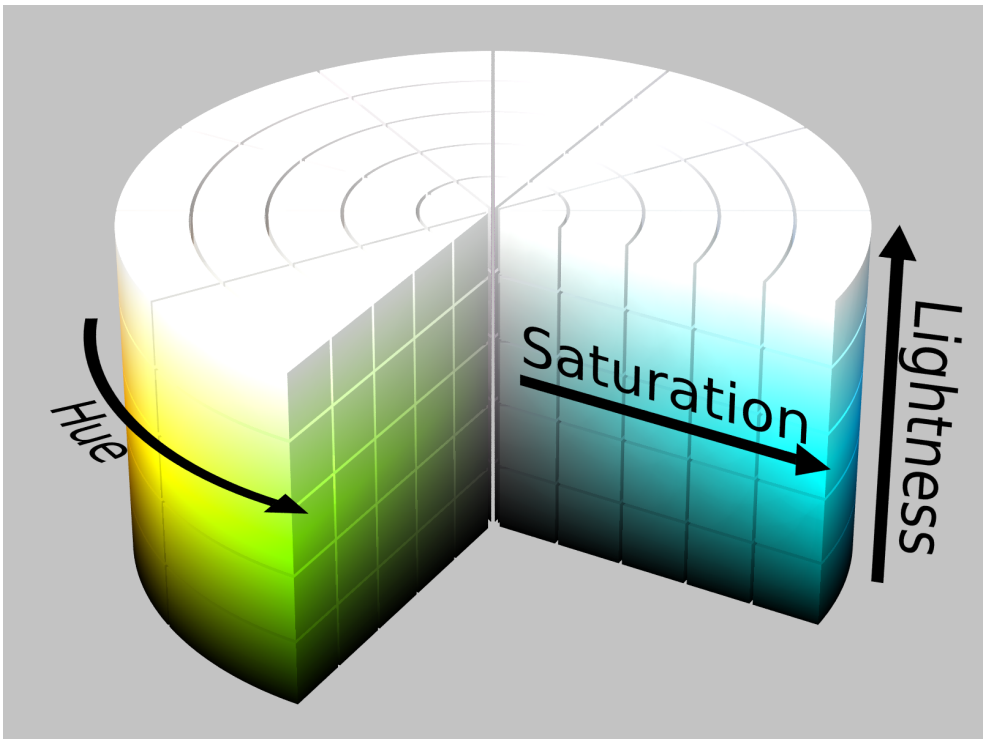

Figure 1: HSL cylinder

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.

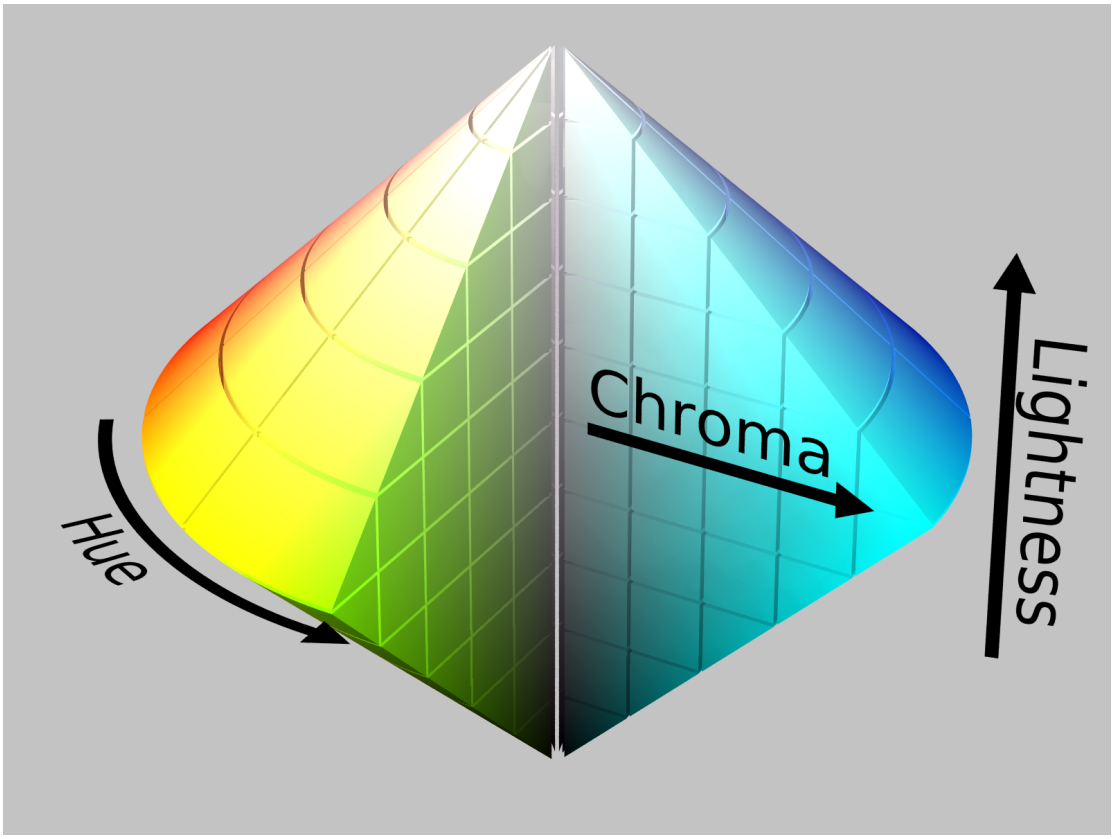

Figure 2: HSL Cone

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 any similar 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.

$$M = \max(r, g, b)$$
$$m = \min(r, g, b)$$
$$c = M - m$$

Then, the lightness is defined as

$$l = (M + m)/2$$

The hue has an awkward piecewise definition as

$$\mathrm{h} = \begin{cases} \text{undefined}, & \text{if } c = 0 \\ 60^\circ\,(g - b)\,/\,c, & \text{if } c \neq 0,\ M = r \text{ and } g - b > 0 \\ 60^\circ\,(b - r)\,/\,c + 120^\circ, & \text{if } c \neq 0,\ M = g \\ 60^\circ\,(r - g)\,/\,c + 240^\circ, & \text{if } c \neq 0,\ M = b \\ 60^\circ\,(g - b)\,/\,c + 360^\circ, & \text{if } c \neq 0,\ M = r \text{ and } g - b < 0 \end{cases}$$

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

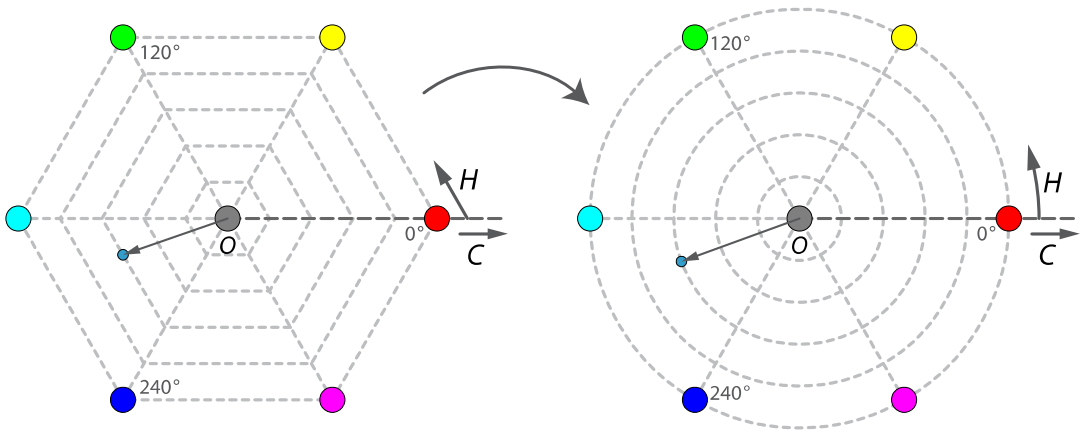

Figure 3: Warping Hexagon into Circle

The saturation is also defined piecewise, as

$$s = \begin{cases} 0, & \text{if } l = 0 \text{ or } l = 1 \\ c\,/\,(1 - |2l - 1|), & \text{if } l \neq 0 \text{ and } l \neq 1 \end{cases}$$

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

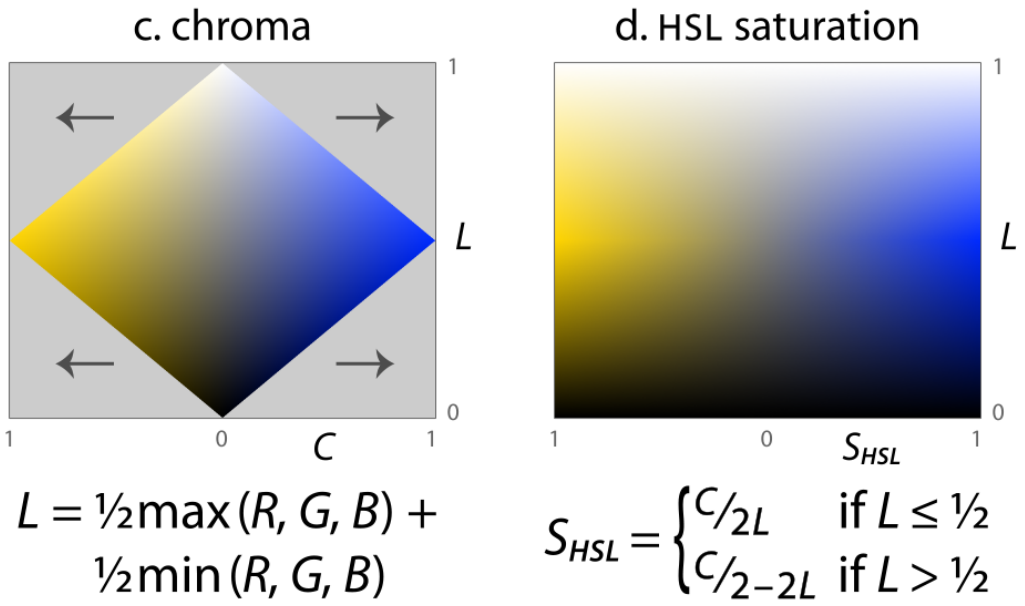

Figure 4: Scaling to Fill the Interval

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. The warp of hue is relatively small, while 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 for two reasons. First, a general RGB model does not give what red, green and blue actually are, as well as their relationships. Second, it is irrelevant to our coordinates transforation.

(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, distortion is not tolerated, including forcing primaries and secondaries into one plane, warping hexagons into circles, and inequal scaling. Hence, 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 formulae for any given point entry (r, g, b) in the RGB space to (h, c, i) in the Improved HCI space, where

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

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.

$$i = (r + g + b)/3$$

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

$$0 \le i \le 1$$

Since the projection preserves the property of 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 Plane Set

The chromatic plane in our model refers to the plane on which every entry has the same intensity i. By this definition, to find a chromatic plane, the cube is not projected; it is clipped instead.

In the RGB space, there is a set of such planes all parallel to each other, whose normal vectors are all (1, 1, 1). In the tilted cube model their normals are all vertical. The neutral axis is one of the normals in both cases.

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.

We name the chromatic plane corresponding to $i = 0.5$ as the base plane, which is the only regular hexagon among the set.

The center of every chromatic plane is on the neutral axis. We may call it neutral point. It is (i, i, i) in an RGB space.

These concepts are important when defining the following parameters, since we must work with these planes to obtain the results.

3.5. Deviation and Chroma

Deviation, denoted d, is the prototype of another parameter, chroma. We define it as the distance from the point to the neutral axis:

$$d = \text{sqrt}\ [(r - i)^2 + (g - i)^2 + (b - i)^2]$$

Alternatively,

$$d = \text{sqrt}\ \{1/3 * [(r - g)^2 + (g - b)^2 + (b - r)^2]\}$$

The latter form is favorable due to its independence of intensity, yielding better interoperatability in case the definition of intensity is substituted.

The result lies in the range $0 \le d \le \sqrt{6} / 3$. To obtain the chroma c, we stretch it linearly to fill the interval from 0 to 1:

$$\begin{aligned} c &= (\sqrt{6} / 2)\ d \\ &= \text{sqrt}\ \{1/2 \cdot [(r - g)^2 + (g - b)^2 + (b - r)^2]\} \end{aligned}$$

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

We find some uses to have the shape stretched farther to the condition where it is safe for every entry on the base plane to be assigned a chroma between 0 and 1. In other words, we draw an internal tangent circle for the base plane, and scale it linearly so that the radius of this circle is 1. In this version, we have

$$\begin{aligned} c' &= (2\sqrt{6} / 3)\ d \\ &= 4/3 \cdot \text{sqrt}\ \{1/2 \cdot [(r - g)^2 + (g - b)^2 + (b - r)^2]\} \end{aligned}$$

Both c and c' are reasonable definitions. Which one wins utility remains to be seen.

3.6. Hue

Lastly, we define the hue h if the entry is not on the neutral axis. On the chromatic plane, we draw vector v_1 from the neutral point to the direction of pure red and v_2 from the neutral point to the entry. The hue h is defined as the angle from v_1 to v_2. One can easily write

$$\cos h = (1, -1, -1) \cdot (r - i, g - i, b - i) / (\sqrt{3} \cdot d)$$

Solving the equation, we obtain

$$\begin{aligned} h = \ & \arccos [(2r - g - b) / 2c], && \text{if } g - b \ge 0 \\ & 360° - \arccos [(2r - g - b) / 2c], && \text{if } g - b < 0 \end{aligned}$$

Hence,

$$\begin{aligned} h = \ & \text{undefined}, && \text{if } c = 0 \\ & \arccos [(2r - g - b) / 2c], && \text{if } c \ne 0,\ g - b \ge 0 \\ & 360° - \arccos [(2r - g - b) / 2c], && \text{if } c \ne 0,\ g - b < 0 \end{aligned}$$

An alternative expression is in terms of arctangent. The algebraic expression is more simple, but with 6 cases in total. We omit it here.

3.7. Summary

In summary, the parameters for Improved HCI model are intensity i, chroma c, hue h. The formulae are:

$i = \quad (r + g + b)/3$

$$
\begin{aligned}
d &= \text{sqrt}\,\{1/3 \cdot [(r-g)^2 + (g-b)^2 + (b-r)^2]\} \\
c &= \sqrt{6}/2\,d \\
h &= \begin{cases}
\text{undefined}, & \text{if } c = 0 \\
\arccos\,[(2r-g-b)/2c], & \text{if } c \neq 0,\ g-b \geq 0 \\
360^\circ - \arccos\,[(2r-g-b)/2c], & \text{if } c \neq 0,\ g-b < 0
\end{cases}
\end{aligned}
$$

4. BACKWARD CONVERSION

To find the conversion from Improved HCI to RGB, we solve the equations backwards. We have

$$
\begin{aligned}
r &= i + 2/3 \cdot c \cdot \cos h \\
g &= i + 2/3 \cdot c \cdot \cos (h - 120^\circ) \\
b &= i + 2/3 \cdot c \cdot \cos (h - 240^\circ)
\end{aligned}
$$

if the hue is defined.

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

$$r = g = b = i$$

5. APPLICATIONS

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

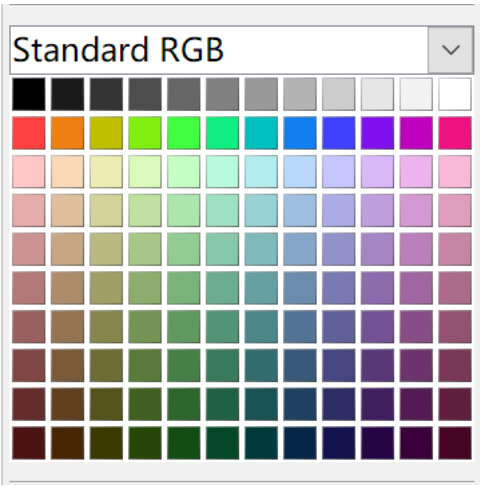

Figure 5: Palette Created with Improved HCI

The first row contains a grayscale with intensity $i = 0.00, 0.10, 0.20, \ldots, 0.90, 1.00,$ plus 0.95.

The second row contains highly saturated colors with hue $h = 30^\circ n$, n being a natural number, chroma $c = 0.75$ ($c' = 1$), 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 $i = 0.85, 0.75, \ldots, 0.25, 0.15$.

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 CONVERTOR 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.

A.2.1. Forward conversion

```
/* Copyright 2018 Flora Canou, Alexander Zheng | V. C0-C1 (1.0.1)
| RGB to Improved HCI Convertor
 * This Source Code Form is licensed under the Mozilla Public Li-
cense, v. 2.0.
 * If a copy of the MPL was not distributed with this file, you
can obtain one at https://mozilla.org/MPL/2.0/.
 * The program converts an entry in an RGB color space to the
corresponding Improved HCI representation.
 * See Improved RGB Color Space Cylindrical Representation for
more info.
 */

#include <stdio.h>
#include <math.h>
#define pi    3.14159265
```

```
double findHue (double r, double g, double b, double c)
{
    double h_rad, h_deg; //hue in radians, hue in degrees
    if (r == g && g == b) //gray, hue is undefined
    {
        return -1;
    }
    else
    {
        if (g > b)
            h_rad = acos ((2 * r - g - b) / (2 * c));
        else if (g < b)
            h_rad = 2 * pi - acos ((2 * r - g - b) / (2 * c));
        else //g == b has to be given saparately, otherwise produc-
ing error
            h_rad = 0;
        h_deg = h_rad * 180 / pi;
        return (h_deg);
    }
}

void convertForth (double r, double g, double b, double *i, dou-
ble *c, double *h)
{
    double d; //deviation
    *i = (r + g + b) / 3;
    d = sqrt (((r - g)*(r - g) + (g - b)*(g - b) + (b - r)*(b -
r)) / 3);
    *c = d * sqrt (6) / 2;
    *h = findHue (r, g, b, *c);
}

void instruct (void);
int main (void)
{
    instruct();
    double r, g, b; //red, green, blue
    double i, c, h; //intensity, chroma, hue in degrees

    while (1)
    {
        printf ("Please enter the values. > ");
        scanf ("%lf %lf %lf", &r, &g, &b);
        if (r == -1 && g == -1 && b == -1)
            return 0;
```

```
       convertForth (r, g, b, &i, &c, &h);
       if (h == -1)
          printf ("Intensity\t= %f, \nChroma\t\t= %f, \nHue is
undefined. \n", i, c);
       else
          printf ("Intensity\t= %f, \nChroma\t\t= %f, \nHue\t\t=
%f degrees. \n", i, c, h);

       if (r < 0 || g < 0 || b < 0 || r > 1 || g > 1 || b > 1)
          printf ("Warning: One or more values are out of range.
The result may be erroneous. \n");
   }
}

void instruct (void)
{
   printf ("This program converts an entry in an RGB color space
to the corresponding Improved HCI representation. \n"
          "You will be asked to enter the values for red, green
and blue, respectively. \n"
          "They should be between 0 and 1 and be separated with a
space. \n"
          "Enter -1 -1 -1 to exit. \n\n");
}
```

A.2.2. Backward Conversion

```
/* Copyright 2018 Flora Canou, Alexander Zheng | V. C0-C1 (1.0.1)
| Improved HCI to RGB Convertor
 * This Source Code Form is licensed under the Mozilla Public Li-
cense, v. 2.0.
 * If a copy of the MPL was not distributed with this file, you
can obtain one at https://mozilla.org/MPL/2.0/.
 * The program converts an entry in an Improved HCI color space
to the corresponding RGB representation.
 * See Improved RGB Color Space Cylindrical Representation for
more info.
 */

#include <stdio.h>
#include <math.h>
#define pi    3.14159265

void convertBack (double h_deg, double c, double i, double *r,
double *g, double *b)
```

```
{
    double h_rad; //hue in radians
    h_rad = h_deg * pi / 180;
    *r = i + 2 * c * cos (h_rad) / 3;
    *g = i + 2 * c * cos (h_rad - 2 * pi / 3) / 3;
    *b = i + 2 * c * cos (h_rad - 4 * pi / 3) / 3;
}

void instruct (void);
int main (void)
{
    instruct();
    double h, c, i; //hue in degrees, intensity, chroma
    double r, g, b; //red, green, blue

    while (1)
    {
        printf ("Please enter the values. > ");
        scanf ("%lf %lf %lf", &h, &c, &i);
        if (h == -1 && c == -1 && i == -1)
            return 0;

        convertBack (h, c, i, &r, &g, &b);
        printf ("Red\t= %f, \nGreen\t= %f, \nBlue\t= %f. \n", r, g,
b);

        if (h < 0 || c < 0 || i < 0 || h > 360 || c > 1 || i > 1)
            printf ("Warning: One or more values are out of range.
The result may be erroneous. \n");
    }
}

void instruct (void)
{
    printf ("This program converts an entry in an Improved HCI
color space to the corresponding RGB representation. \n"
            "You will be asked to enter the values for hue, chroma
and intensity, respectively. \n"
            "The hue should be in degrees, between 0 and 360. The
other two should be between 0 and 1. \n"
            "Note: In case the hue is undefined, enter any value
for hue and 0 for chroma. \n"
            "All of them should be separated with a space. Enter -1
-1 -1 to exit. \n\n");
}
```