INTERNATIONAL LASER
DISPLAY ASSOCIATION

Technical Committee

Implementing the ILDA Standard Palette

Revision 001, September 1995

Implementing the ILDA Standard Palette

Why is an ILDA Standard Palette necessary?

The ILDA Standard Palette was developed to fix color-handling limitations in the original ILDA File Format interchange standard. The File Format allows one company to send a laser graphic frame to another company.

The ILDA File Format works well for transferring coordinate data: X, Y and Z values. But it does not work well for transferring color data: red, green and blue values. In the ILDA File Format, each point has a color number, but the colors themselves were undefined. One system may use "color 1" to mean "white", while another system uses it to mean "red". The result is that colors change when files go from one system to another.

In the first attempt to solve this, a color palette section was added to the ILDA File Format. A well-defined set of rules determined how to transfer custom color palettes. Unfortunately, the rules were relatively complex. As of this writing (fall 1995), no companies have implemented the color palette section of the ILDA File Format.

Therefore, another proposal was set forth. The ILDA Standard Palette assigns RGB values to the previously arbitrary color numbers. For example, ILDA Color 1 is now defined as: red=255, green=255, blue=255. This color is, of course, white. (Values are expressed on a 0 to 255 scale: 0 is off [0 volts] while 255 is full on [+5 volts].)

The ILDA Standard Palette allows systems to know what color to read and write, so that the 256 closest-matching colors are transferred.

Important Concepts

There are some important concepts which you should understand:

- Having a standard palette does not mean that laser computer systems have to use this palette internally. Laser systems can use whatever color information or method they want. The ILDA Standard Palette merely helps different systems to exchange frames and animations.

- You are not restricted to the colors in the ILDA Standard Palette. If you must transfer more than 256 colors, or if you must use colors that are not defined in the ILDA Standard Palette, you can still utilize the custom color palette section of the ILDA File Format. The ILDA Standard Palette is a merely a simpler alternative. It is intended for the majority of systems with relatively simple color needs.
- ILDA colors are idealized. For example, they do not take into account that krypton red is different from HeNe red. At this time, they do not take into account that some color devices (e.g. PCAOMs) are usually non-linear. It is assumed that your hardware or software has controls to make the laser output come as close to the idealized ILDA color as possible.

What If You Use a Custom Palette?

As discussed above, the ILDA File Format specification includes a method which allows you to specify your own custom color palette. Should you decide to implement this, you will not be able to simultaneously support the ILDA Standard Palette. In other words, a single file cannot use both a custom palette and the ILDA palette. This is because the custom palette numbers and values will, by definition, be different from ILDA's.

In such a case, you may want to give the user a choice: save frames either using a custom palette or using ILDA's palette. The custom palette file can be sent to those with software that implements this section of the ILDA format. The ILDA palette file can be sent to all others.

Implementing the ILDA Standard Palette

There are two types of laser computer graphics systems:

- Systems using color palettes. Each point has a “color number”. This is used as an entry to a lookup table to find what RGB (red, green and blue) values to use for that point. Most laser systems use palettes.

- Systems using “true color”. Each point has its own RGB value. No palette or lookup table is necessary.

Implementation of the ILDA Standard Palette works best with systems that use color palettes. We'll discuss these first, then move on to implementation on systems that use true color.

Transferring Color Data Using Color Palette Systems

Consider the common task of transferring color data from a laser graphics system using one color palette, to a laser graphics system using a completely different palette. We'll call the first system the “source”, and the second the “destination”.

The solution to the problem is to use the ILDA Standard Palette as an intermediate step. The source system translates its color numbers into the closest Standard Palette color numbers. The destination system then translates the Standard Palette color numbers into its closest color numbers. Here is an example that uses just four colors, for simplicity:

Source Palette

Color number	*Red value*	*Green value*	*Blue value*	*Closest ILDA color number*
1	125	75	10	146
2	50	0	0	209
3	250	100	0	79
4	255	255	255	1
Note: Color values in all three palettes are expressed as 8-bit numbers, between 0 and 255				

ILDA Standard Palette

Color number	**Red value**	**Green value**	**Blue value**	**Notes**
1	255	255	255	White
79	255	116	0	Orange
146	128	70	0	Dark orange
209	64	0	0	Dark red
Note: Only four entries shown for example purposes				

Destination Palette

Color number	**Red value**	**Green value**	**Blue value**	**Closest ILDA color number**
1	255	255	255	1
2	64	8	8	209
3	128	64	16	146
4	255	128	32	79

The tables show three example palettes:

- a source system's palette
- the ILDA Standard Palette
- a destination system's palette

When writing an ILDA standard file, the source system looks at the internal color number (for example, color 2) and writes out the closest ILDA color (ILDA color 79 in the example).

When reading in an ILDA standard file, the destination system looks at the ILDA color (79) and writes out its closest internal color number (color 4 in the bottom table).

Each laser computer graphics system will internally have two lookup tables. One table converts the system's colors when writing to ILDA format. The other table converts incoming ILDA colors back into the system's colors.

Note that some of the source and destination colors are not exact matches to the ILDA colors. However, they are fairly close — certainly , much better than if there was no ILDA Standard Palette at all.

Here are details on how to convert from the source palette to the ILDA Standard Palette.

Converting From a Palette System to the ILDA Standard Palette

You first need to make a source-to-ILDA lookup table. To use it, you look up your palette color number; the table then tells you the closest match in the ILDA Standard Palette.

If you have a simple color system — for example, there are only eight total colors — simply make the source-to-ILDA table by hand. For each color you have, look in the table of ILDA Standard Palette colors to find the closest match.

If you have more palette slots, it is probably easiest to write a short program to find the closest colors. (Note: Color palette data can be found on the diskette included in the ILDA Handbook, or contact ILDA for a spreadsheet-type file. They cannot easily be computer-generated.) The program compares your colors to the hard-coded ILDA values. For every one of your colors, the program calculates the “shortest distance” to the ILDA colors. The following pseudo-code shows how to do this:

```
• Initialize an array of integers called “source2ILDA”. It has the same
  number of elements as you have color numbers in your palette.

• Initialize an array of long integers called “differencearray”. It has 256
  elements.

• For each color number in your palette

    • Get the red, green and blue values of that color number

    • For each color number in the ILDA Standard Palette (256 colors)

        • Get the red, green and blue values of that ILDA color number

        • Compute the red, green and blue differences as follows:
          reddifference = ILDAredvalue - yourredvalue
          greendifference = ILDAgreenvalue - yourgreenvalue
          bluedifference = ILDAbluevalue - yourbluevalue
```

        - Square the differences and sum them as follows: differencescore = reddifference^2 + greendifference^2 + bluedifference^2

        - Store the difference score in the “differencearray” (array index =ILDA color number)

    - Get next ILDA color number. (At this point, you have an array of 256 difference scores — one for each color in the ILDA Standard Palette. The closer the match, the smaller the score.)

    - Initialize a variable called “smallestdifference” to 195,076.

    - Initialize a variable called “closestmatch” to 0.

    - For each entry in “differencearray” (256 entries)

        - If the difference score is less than “smallestdifference,” then set “smallestdifference” to the new (lower) difference score. Also, remember which entry was the lowest in “closestmatch.”

    - Get next entry. (At this point, the value of “closestmatch” indicates the ILDA Standard Palette color that is the closest match to the color number's RGB values.)

    - Store “closestmatch” in the “source2ILDA” array (array index = your color number).

- Get next color number in your palette.

Notes

- This algorithm can of course be optimized further. For example, if an exact match is found, no further comparisons to ILDA colors are necessary, and you can move on to the next color number in your palette.

- The worst possible difference score is found by comparing white (255, 255, 255) to black (0, 0, 0): $255^2 + 255^2 + 255^2$, or 195,075. This is why the value 195,076 is initialized in the variable “smallestdifference” — all subsequent comparisons are likely to have a lower score than this.

- If your system uses color cycling, you may want to differentiate between colors which are “fixed” and those which are “cycleable”. It is probably easiest to do this by hand, once the computer has made the basic source-to-ILDA lookup table.

The result of this operation is a lookup table: the “source2ILDA” array. The array indexes, of course, will be the source system's color numbers: 1, 2, 3, 4...etc. The array contents will be the closest ILDA Standard Palette color number: for example, 146, 209, 79, 1...etc.

When you write an ILDA format file, get the color number for each point and use the lookup table to find the corresponding ILDA color number. In the example above, if the source point's color number is 1, look in the contents of array element 1. You'll find the closest ILDA color number, which is 146. Write out this number into the ILDA Format file.

A significant advantage of the lookup table method is that you need to create the table just once. You can then hardcode it into your ILDA file conversion code, or you could store it on disk and load it as needed. Once you have the table, it is a simple, fast procedure to convert from your palette to the ILDA Standard Palette.

If you use more than one palette, you'll need to make a lookup table for each palette.

Converting from the ILDA Standard Palette to a Palette System

To convert colors back from the ILDA Standard Palette to your destination palette, make a similar lookup table. This one starts with the ILDA Standard Palette, and finds the closest match in your palette. The code will be almost identical to the previous pseudo-code.

As with the source-to-ILDA lookup table, the ILDA-to-destination table only needs to be derived once (for each palette in your system). You can hardcode the table right into your program code.

Transferring Color Data Using “True Color” Systems

It is slightly more complex to convert from a true color system that stores each point's RGB values. Fortunately, it is trivial to go the other way — to convert from ILDA to true color.

Converting From True Color to the ILDA Standard Palette

If you have a true color system, you'll need to determine the closest ILDA Standard Palette on a point-by-point basis. It isn't practical to precompute a lookup table, because the table would have 16.7 million entries (on an 8-bit RGB system).

To find the closest ILDA match on a point-by-point basis will unfortunately take some computer time. Remember, however, that the ILDA format is for image interchange only. Even if the conversion takes a while, this is only done once when you convert the frame to ILDA format. It does not affect your system's native frame storage time.

The code to perform the conversion is very similar to that used by the source-to-ILDA program. Here is a pseudo-code version:

```
• Initialize a long integer array called “differencearray”. It has 256 elements.
• For each point in your frame
    • Get the red, green and blue values of that point
    • For each color number in the ILDA Standard Palette (256 colors)
        • Get the red, green and blue values of that ILDA color number
        • Compute the red, green and blue differences as follows:
        • reddifference = ILDAredvalue - yourredvalue
          greendifference = ILDAgreenvalue - yourgreenvalue
          bluedifference = ILDAbluevalue - yourbluevalue
        • Square the differences and sum them as follows: differencescore =
          $\text{reddifference}^2 + \text{greendifference}^2 + \text{bluedifference}^2$
```

-
  -
    - Store the difference score in the “differencearray” (array index = ILDA color number)
  - Get next ILDA color number (At this point, you have an array of 256 difference scores — one for each color in the ILDA Standard Palette. The closer the match, the smaller the score.)
  - Initialize a variable called “smallestdifference” to 195,076
  - Initialize a variable called “closestmatch” to 0
  - For each entry in “differencearray” (256 entries)
    - If the difference score is less than “smallestdifference”, then set “smallestdifference” to the new (lower) difference score. Also, remember which entry was the lowest in “closestmatch”
  - Get next entry (At this point, the value of “closestmatch” indicates the ILDA Standard Palette color that is the closest match to your point's RGB values.)
  - Write point data to ILDA file, using “closestmatch” as the ILDA color number
- Get next point in your frame.

Notes

- This algorithm can of course be optimized further. For example, if an exact match is found, no further comparisons to ILDA colors are necessary, and you can move on to the next point in your frame.
- Note that the worst possible difference score is found by comparing white (255, 255, 255) to black (0, 0, 0): $255^2 + 255^2 + 255^2$, or 195,075. This is why the value 195,076 is initialized in the variable “smallestdifference” — all subsequent comparisons are likely to have a lower score than this.
- If your system uses color cycling, you may want to differentiate between colors which are “fixed” and those which are “cycleable”. Since there are only eight dual-purpose (fixed/cycleable) colors, you could have a special-case section after the initial conversion is done.

Converting From the ILDA Standard Palette to True Color

Fortunately, it is trivial to make an ILDA-to-destination RGB table. Merely store the ILDA Standard Palette! As you get each point's ILDA color number, look in the Standard Palette to find the RGB values for that point.

From the Standard to the Screen

You may have noticed that the ILDA Standard Palette is not referenced to any colorimetric standard. For example, the color red is defined as RGB values 255, 0, 0. This means full red (on a 0 to 255 scale), no green and no blue. But nowhere is the red wavelength defined.

This is because the deep red produced by a krypton laser is quite different from the orangish-red produced by a helium-neon laser. Yet both colors are the “reddest” which the lasers can produce. At this time, laser technology is not standardized or advanced enough to specify a particular mix of wavelengths.

RGB Not Defined

For this reason, ILDA does not currently define what is meant by “red”, “green” and “blue”. Generally, they will be the “reddest”, “greenest” and “bluest” colors available from the laser source(s). In some cases, you may need to make color-shift adjustments when transferring frames from systems with laser sources that have very different wavelengths than yours.

RGB Assumed to be Linear

ILDA also assumes that the projected RGB colors will be linear. That is, if you draw a series of gray-scale lines, the grays should be evenly spaced. Unfortunately, many color (or intensity) devices are non-linear. For example, in most PCAOM drivers, all the dark grays may look black, the medium grays will rapidly rise from black to white, and the light grays will look white.

Due to the wide variety of color/intensity devices, ILDA colors do not correct for non-linearities. The ILDA Standard Palette assumes your colors are linearized either through hardware or through additional software settings.

For example, some laser software allows different settings for the screen colors (the ones you draw with on the computer monitor) and the laser colors (the ones you see projected). The ILDA RGB settings should be used for the screen colors. (This assumes the computer monitor is linear!) You then set the laser colors so they best match the screen colors. Since you are setting the laser by eye, you will automatically compensate for any non-linearities in your color device.

Multiple Channels (PCAOMs) Don't Matter

Until about 1993, most color laser projectors used three channels of color: red, green and blue. In 1993, multiple-channel PCAOMs became common. They can provide up to eight channels of color control. For example, a single PCAOM might control two different red wavelengths, a yellow, two greens, two blues, and a violet.

Fortunately, the number of output channels should not affect the ILDA Standard Palette. This is because the ILDA colors are “idealized” colors. The specific output device simply tries to come as close as possible to the “idealized” color.

The advantage of more than three channels is that it may be possible to come closer to a desired color. For example, a particular deep purple might be more easily attained using just the violet wavelength.

If your equipment can control more than just three color channels, you will probably have to put up a test pattern and set each output color to match the “idealized” color. Fortunately, this has to be done just once for each projector/laser combination you use. (Note that as laser power changes, the color balance also may change, so you may also have to compensate if you use different power levels, or as the laser ages.)

Questions and Answers

Why is there an entry in the ILDA Standard Palette for black = 0, 0, 0? I thought that the ILDA File Format has a blanking bit for each point.

Any blanked point should be written using ILDA Standard Palette color 0, which has RGB values of 0, 0, 0 (black). The point should also have the blanking bit set in the ILDA file's Status Byte. Having the same information — that the point is blanked — encoded two different ways gives some flexibility. For example, systems that use only RGB data (with no separate blanking device) will not have to perform any special conversions.

How does the color matching algorithm work?

The color matching algorithm takes the RGB differences, squares them, and then adds them. It is essentially using the Pythagorean Theorem to find the shortest distance in 3D color space. (Computing the square root of the sum of the squares takes time and is not necessary. The algorithm does not care about the actual distance, only which distance is shortest.)

You are free to use any other algorithm if you wish. The goal is simply to set up tables to convert your colors to (and from) the closest matching ILDA colors.

There are only 256 color combinations in the ILDA Standard Palette, and yet 8-bit RGB systems can have over 16.7 million color combinations. Isn't 256 colors limiting?

Most laser companies use fairly limited palettes of 8, 16, 64 or 256 colors. Few, if any, laser systems use more than 256 colors in any one frame or animation. The ILDA Standard Palette contains a range of the most common colors used in laser display. Also, remember the purpose of the ILDA File Format is basic frame interchange. Even if colors cannot be exactly transferred, they are likely to be fairly close, so that minimal touch-up is necessary.

Finally, if you must have an exact match, you can use the color palette section of the ILDA File Format. As of mid-1995, no system is known to be using this. However, should the color palette section become implemented in one or more systems, be sure that both ends of the distribution chain — the source ILDA encoder and the destination ILDA decoder — can properly handle the color palette section.

What happens if I don't use the ILDA Standard Palette?

The same thing that happened before the ILDA Standard Palette was adopted: frames lose their color information as they are transferred between systems. If you don't mind this (for example, if you work only with a single-color system) then you don't need to use the ILDA Standard Palette. Conversely, if you have no standard palette in your system, you may wish to consider adopting the ILDA Standard Palette as the default.

If you do decide to use the custom palette section of the ILDA File Format, be careful. In an ideal world, everyone writing ILDA conversion programs will check for the custom palette, but this may not be the case. There may be some programs which do not check to see if a custom palette is being used. Colors will erroneously be assumed to always use the ILDA Standard Palette.