Antmicro

Raviewer - User Guide

2024-05-15

CONTENTS

1 What is Raviewer? 1
- 1.1 Core features . . . 1

2 Installation 2
- 2.1 Requirements . . . 2
- 2.2 Installation . . . 2

3 Quick start 3
- 3.1 GUI . . . 3
- 3.2 Importing a file . . . 4
- 3.3 Choosing a color format . . . 4
- 3.4 Setting the resolution . . . 5
- 3.5 Controlling color channels . . . 5
- 3.6 Selecting an area . . . 8
- 3.7 Zooming in . . . 8
- 3.8 Color picker . . . 9
- 3.9 Exporting data . . . 9

4 Usage 10
- 4.1 Command line arguments . . . 10
- 4.2 Changing endianness . . . 10
- 4.3 Adding and skipping bytes . . . 12
- 4.4 Reversing bytes . . . 14
- 4.5 Hexadecimal preview mode . . . 16
- 4.6 Theme manager . . . 18
- 4.7 Raw view . . . 19

5 Supported color formats 21
- 5.1 RGB . . . 21
- 5.2 YUV . . . 21
- 5.3 Bayer RGB . . . 22
- 5.4 Grayscale . . . 23
- 5.5 Adding new color formats . . . 23

CHAPTER

ONE

WHAT IS RAVIEWER?

Raviewer is Antmicro's open-source image analysis tool, created to streamline the process of video debugging. It handles arbitrary binary data and visualizes it using selected parameters so that you can quickly and efficiently analyze any image you want.

When your project involves operations like building a camera system, capturing the data with an FPGA board, or implementing camera drivers for new cameras or platforms, the sheer number of moving parts you are juggling before you get usable video makes debugging a difficult process. This is where Raviewer may come in handy and help you accelerate and simplify your product development.

It helped us immensely with some of our projects, like when we built the FPGA debayering core, which included a demosaicing system that changes raw data from CCD or CMOS sensors. Initially, the project was created for our internal needs, but we decided to release it to help reduce frustration related to working with complex engineering problems.

1.1 Core features

Raviewer supports many popular color formats like *RGB, YUV, BAYER, or GRAYSCALE* and lets you add new color formats.

- *Checkboxes controlling the displayed channels*
- *On-click displaying raw data making up a pixel as decoded RGB and YUV*
- *Conversion of the whole or selected part of an image to more complex formats (JPEG, PNG) or raw data*
- *An option to append or remove n bytes from the beginning of the image series*
- *Hexadecimal preview mode*
- *Terminal functionality*
- *Theme manager to adjust font and theme preferences*

CHAPTER

TWO

INSTALLATION

2.1 Requirements

For Raviewer to work, you need to have Python 3.9 or higher installed on your system.

You also need the following Python libraries:

- numpy
- opencv-python
- dearpygui == 1.1.1
- terminaltables
- pytest

Note: Raviewer will automatically download any missing libraries.

2.2 Installation

2.2.1 Arch Linux

```
sudo pacman -Sy python-pip git
pip install git+https://github.com/antmicro/raviewer.git
```

2.2.2 Debian

```
sudo apt-get install python3-pip git python3-pil.imagetk
pip install git+https://github.com/antmicro/raviewer.git
```

CHAPTER

THREE

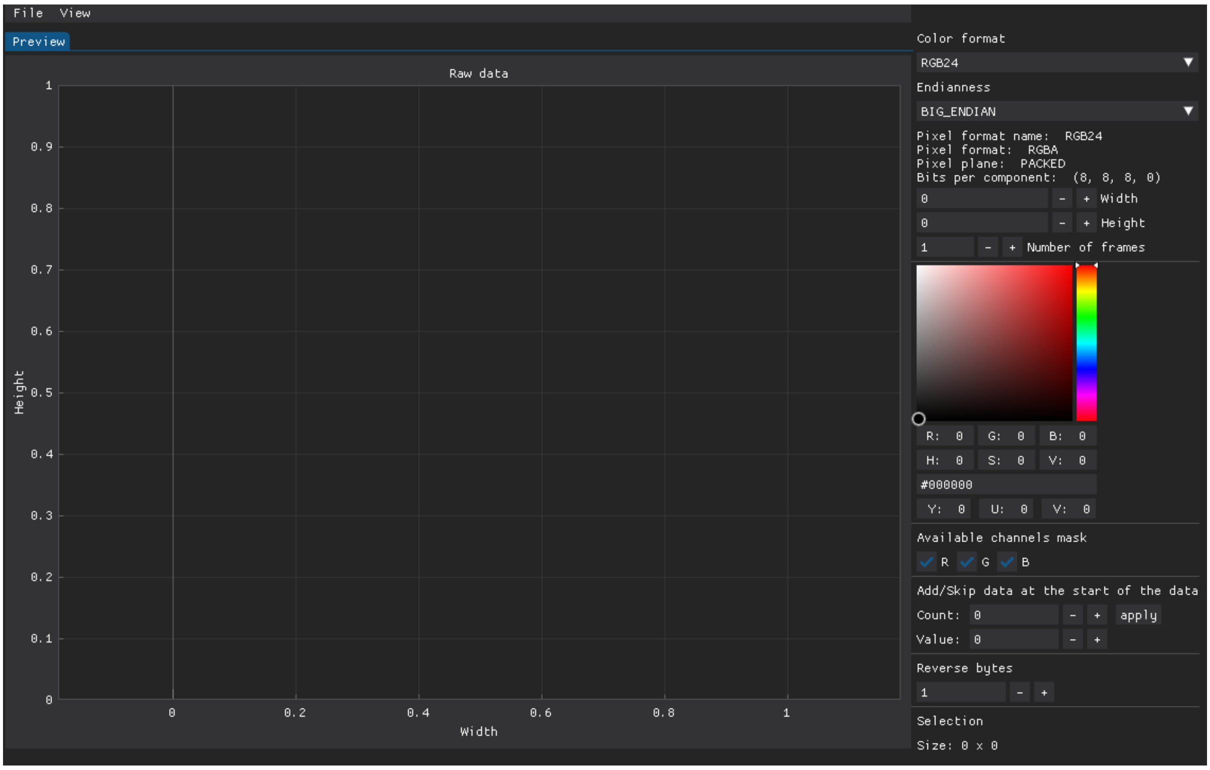
QUICK START

After installing Raviewer, you can start an empty GUI (without any data loaded) using:

```
raviewer
```

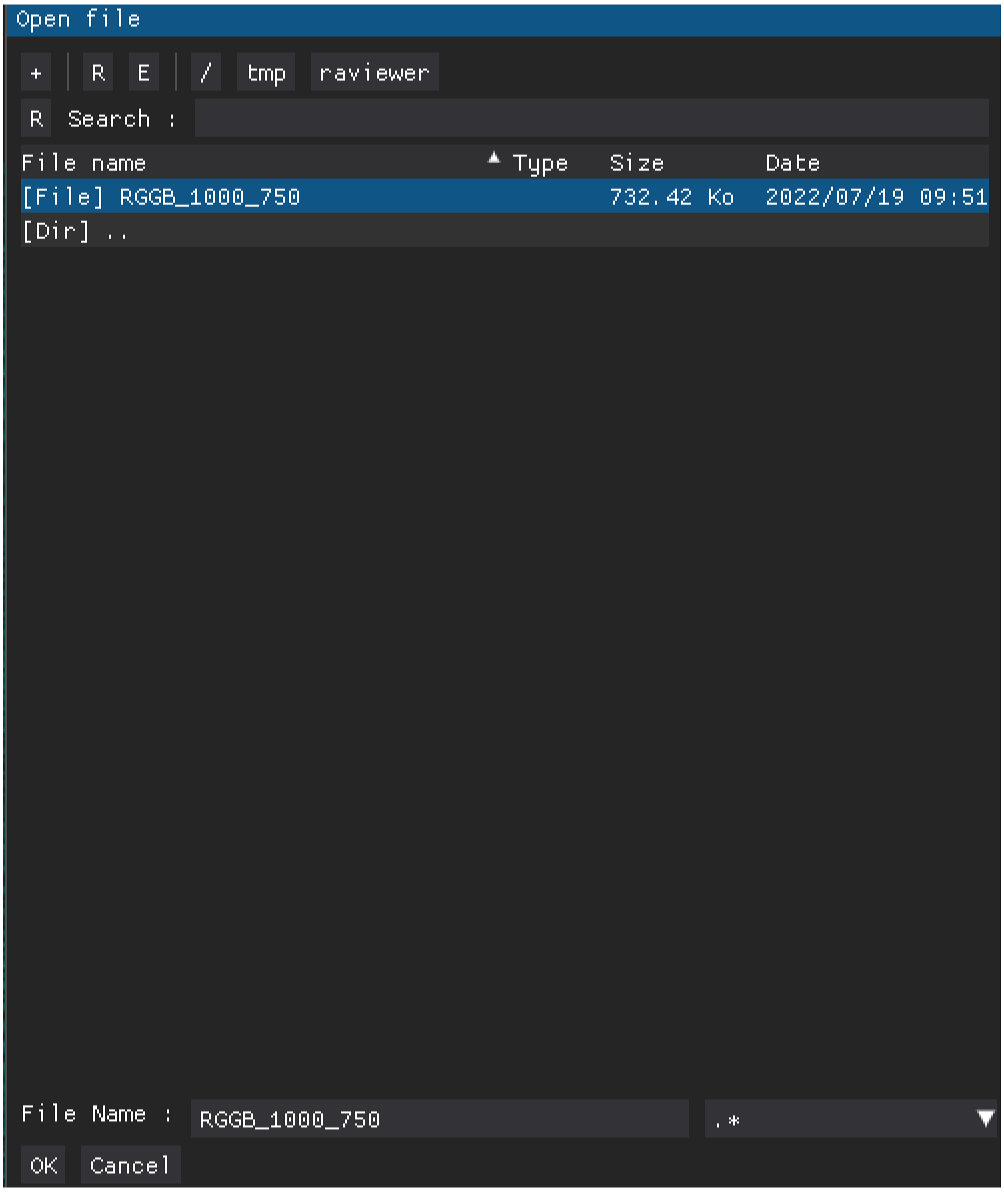
3.1 GUI

Raviewer's GUI is divided into three sections: the work area on the left; properties settings on the right; and the top menu:



3.2 Importing a file

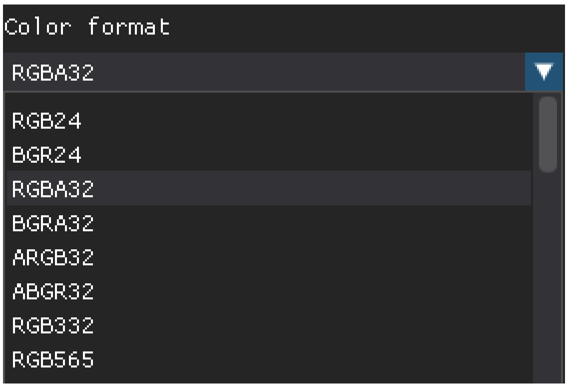
You can import a file to Raviewer by clicking **File** > **Open** from the top menu and selecting the file of your choice:



3.3 Choosing a color format

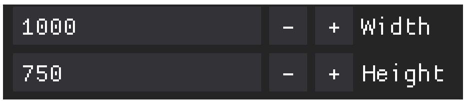
Raviewer supports a vast catalog of color formats, the full list of which you can see in the *chapter devoted to color formats*.

Depending on the sensor you are extracting data from, RAW files can use one of many color formats. You need to select the color format of your file from the **Color format** dropdown menu:



3.4 Setting the resolution

After choosing an appropriate color format for your file, you must adjust the resolution setting to display the preview properly.



The default width in Raviever is 800px, and the height is calculated automatically, so in most cases, you can omit the latter.

Note: You can change the resolution in Raviewer using *command line arguments*, but you must specify the path to the file you want to open for it to work properly.

To open a file with a resolution of 1000x750, you would run:

```
raviewer -w 1000 -H 750 -f /path/to/file
```

The preview should produce a correct, good-looking image if you have set the appropriate color format, *endianness*, and resolution.

3.5 Controlling color channels

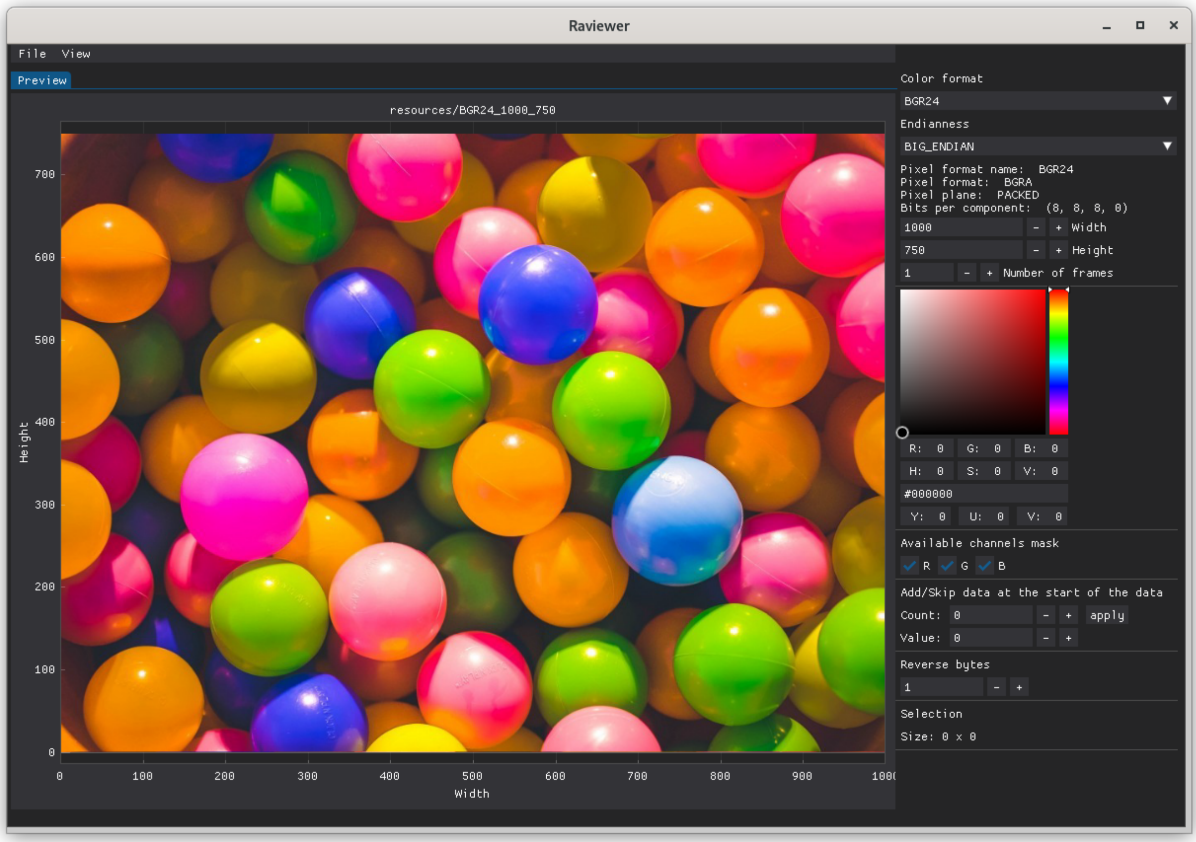
One of the common mistakes you may encounter when working with imported files is the swapped color channels. You can change the displayed color channels to determine if they have been properly assigned.

Note: In some color formats, you can also see if the alpha channel is working correctly.

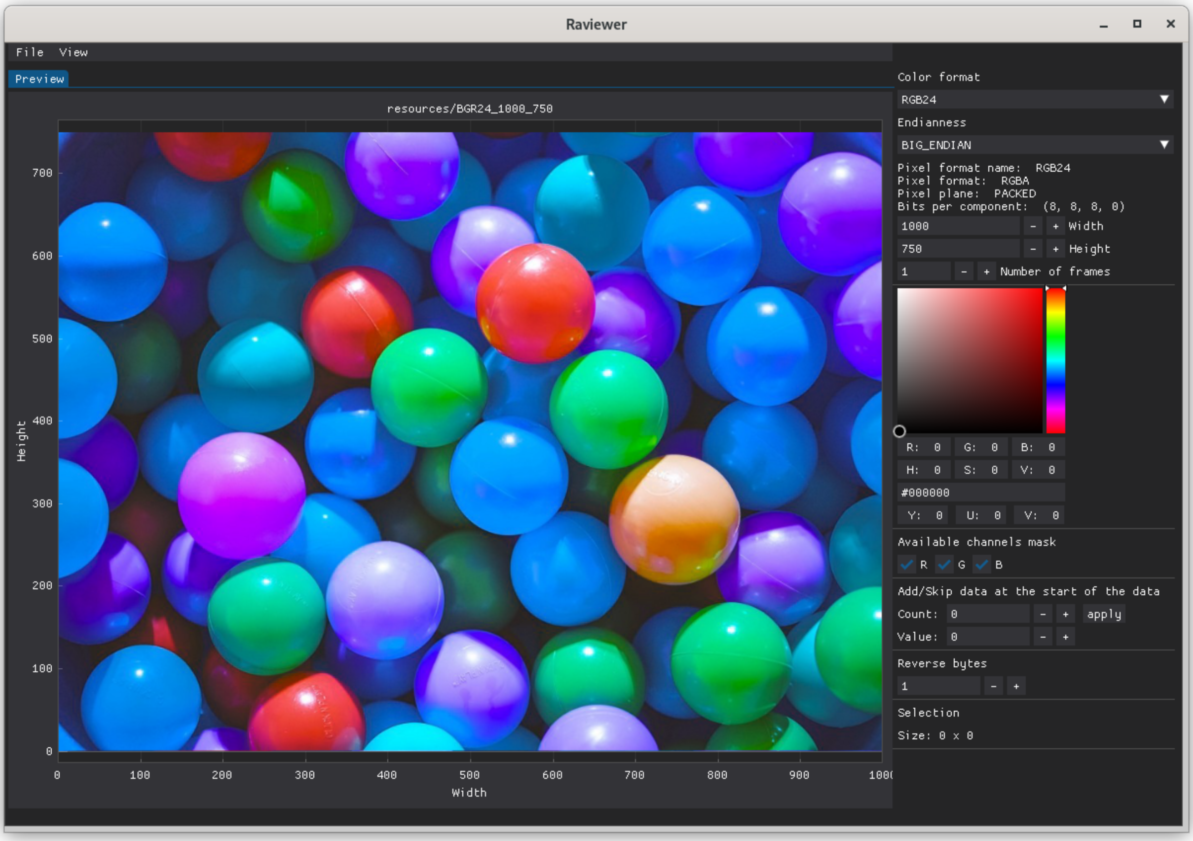
You can easily control which color channel masks are currently being displayed by checking or unchecking boxes on the menu:

Unchecking a box will cause the values of a chosen channel to be set to 0 on every pixel of the picture (except for the alpha channel, which is set to its maximum value).

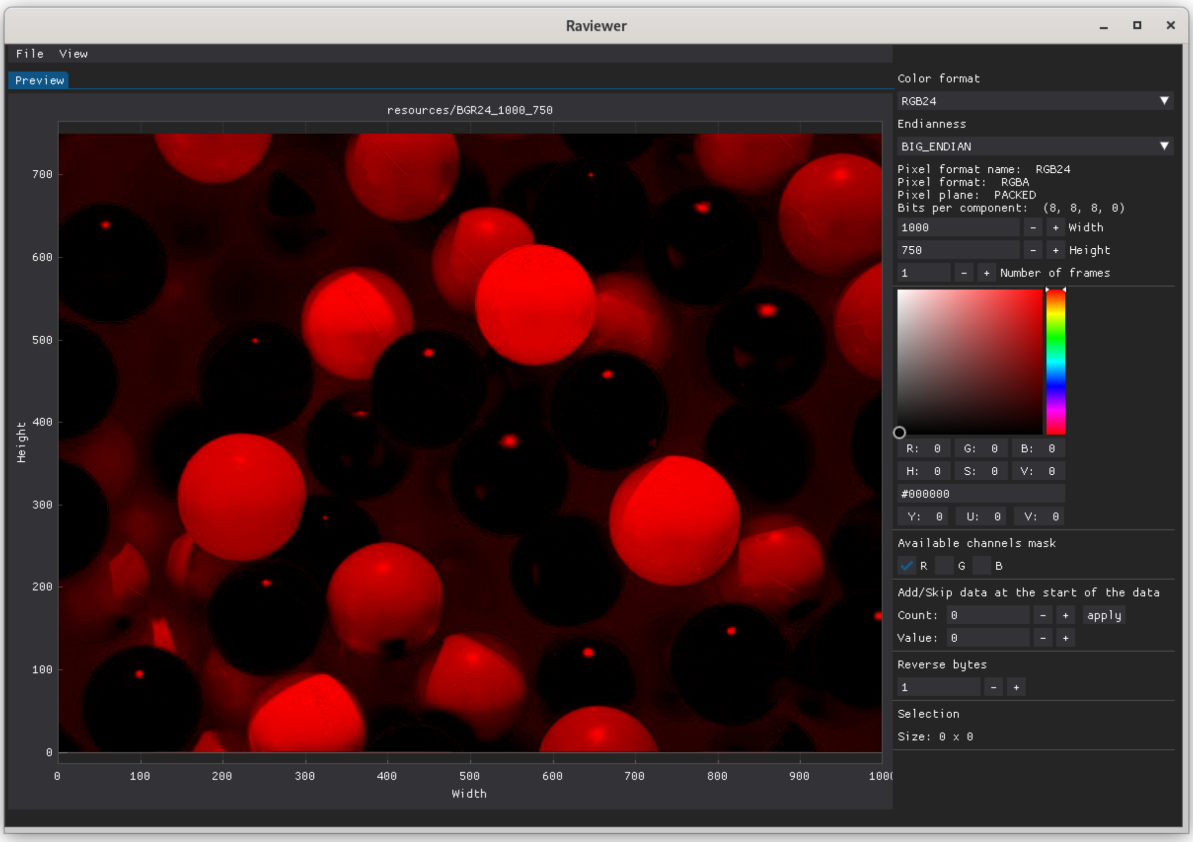
To see how controlling color channels may help you identify issues with your frames, have a look at this BGR24 frame:



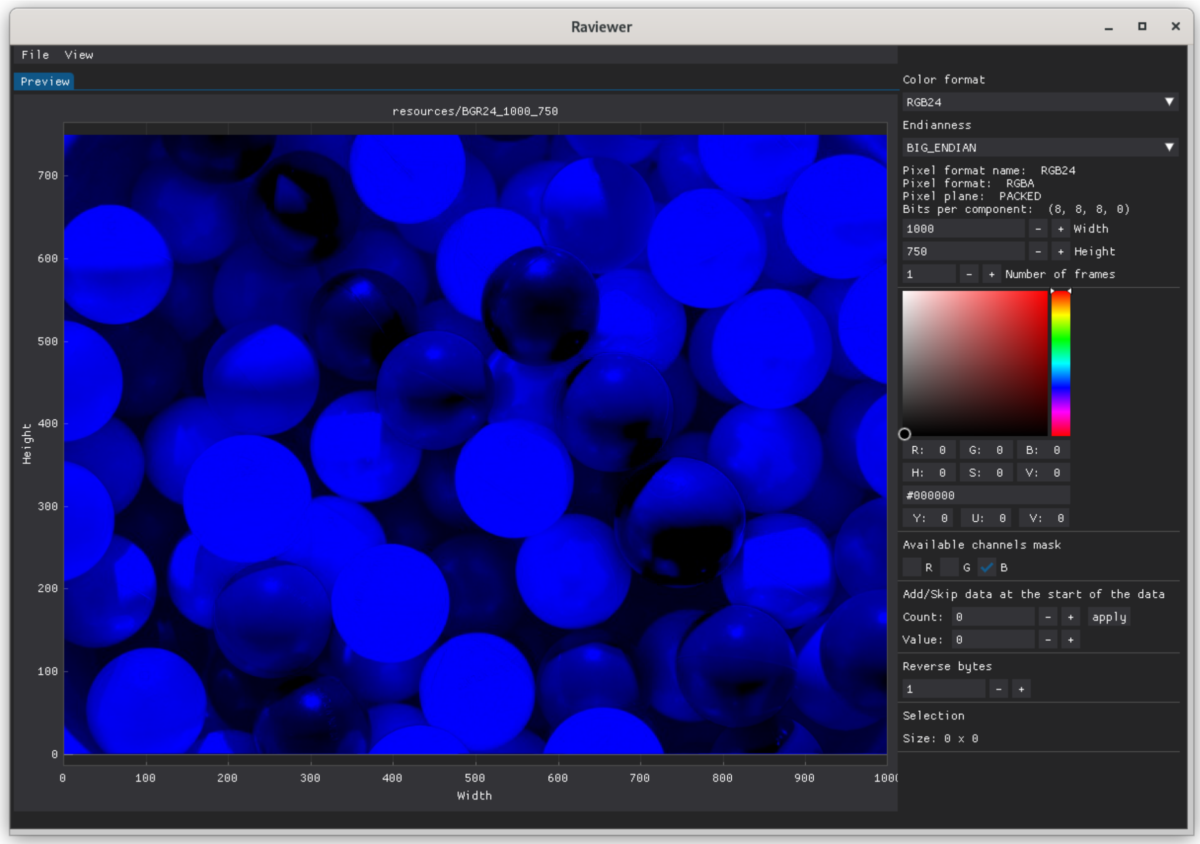
When you select another, wrong color format (in this case, RGB24), you can see that the colors do not match the original ones:



To quickly identify the root of the problem, turn the color channels on and off. When only the red channel is turned on, balls that are blue in reality are shown as very bright:



When only the blue channel is turned on, in reality, orange, pink, or red balls are shown as very bright.



That means that channels R and B are swapped, and the format of our frame is not RGB24 but BGR24.

3.6 Selecting an area

You can select an area of your picture by holding **LMB** (left mouse button) and dragging it over your screen. The selected area will be highlighted in green, and you can see its size in the bottom-right corner.

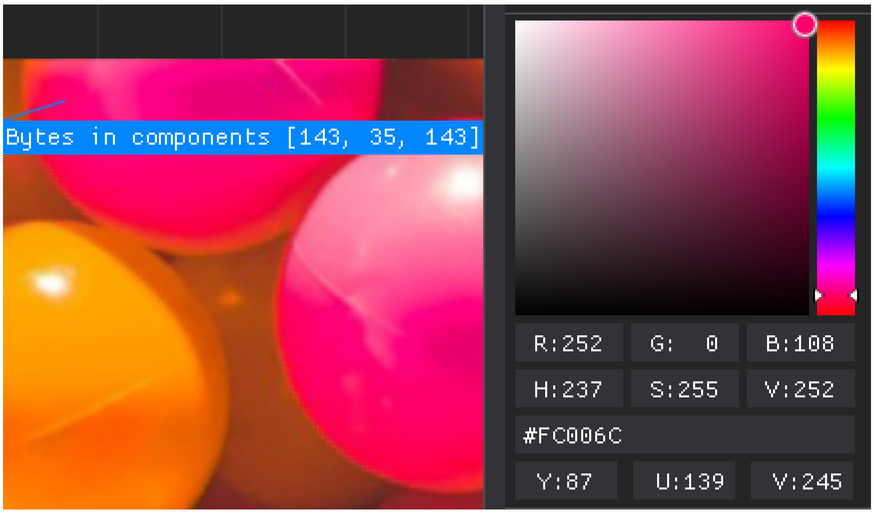
The selected area can then be exported to PNG by selecting **File** > **Export** > **PNG** > **Selection** from the top menu, or to RAW by selecting **File** > **Export** > **RAW** > **Selection**.

3.7 Zooming in

You can zoom in on an area of your picture by holding **RMB** (right mouse button) and dragging it over your screen. The selected area will be highlighted in yellow, and upon releasing **RMB**, Raviewer will fill the whole available workspace with the selected area.

3.8 Color picker

You can display raw data making up a pixel by using **LMB** (left mouse button). It will show you RGB and YUV values as well as the hue, saturation, and lightness of a pixel. You will also be able to see the information about the bytes in a component:



Bytes in components display the channel value in the selected format. You can use this information to determine if the color channel values are correct.

The color picker can help you spot anomalies within your files, like the alpha channel not being set to its maximum value on a picture without transparent elements.

Note: Color channel values in the color picker differ from those on the right side because the former have been converted to their 8-bit RGB counterparts.

The **Bytes in components** window can be closed using **RMB** anywhere in the window.

3.9 Exporting data

Using the top menu, you can export data from Raviewer to PNG or RAW format.

To export a file to PNG, use: **File** > **Export** > **PNG** > **Image**.

To export a file to RAW, use: **File** > **Export** > **RAW** > **Image**.

You can also *export only a snippet of your picture*.

CHAPTER

FOUR

USAGE

This chapter describes more advanced functionalities, but if you want to see how to perform the most basic operations in Raviewer, visit *Quick start chapter*.

4.1 Command line arguments

Raviewer can be launched with already loaded data and parameters (like width or color format). You can find more information about available arguments in command-line help:

```
raviewer --help
```

You can find examples of usage of the available commands in the table below:

Table 4.1: Raviewer's CL arguments

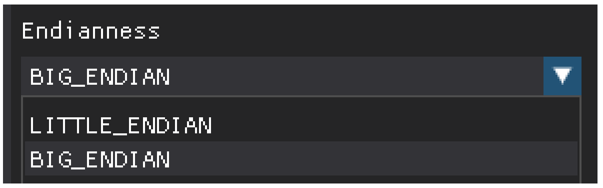
Command	Description
-h (or --help)	Show help message and exit
-c (or --color_format) COLOR_FORMAT	Target *color format* (default: RGB24)
-f (or --FILE_PATH) FILE_PATH	File containing raw image data
-w (or --width) WIDTH	Target width (default: 800)
-H (or --height) HEIGHT	Target height
-e (or --export) RESULT_PATH	Destination file for the parsed image
–list-formats	List available predefined formats
–check-formats	Test all formats

4.2 Changing endianness

The order in which a sequence of bytes is stored can vary in the binary files you import to Raviewer. When working with RAW files, you don't always know your data's format; to determine it, you will have to tweak the endianness and bytes settings.

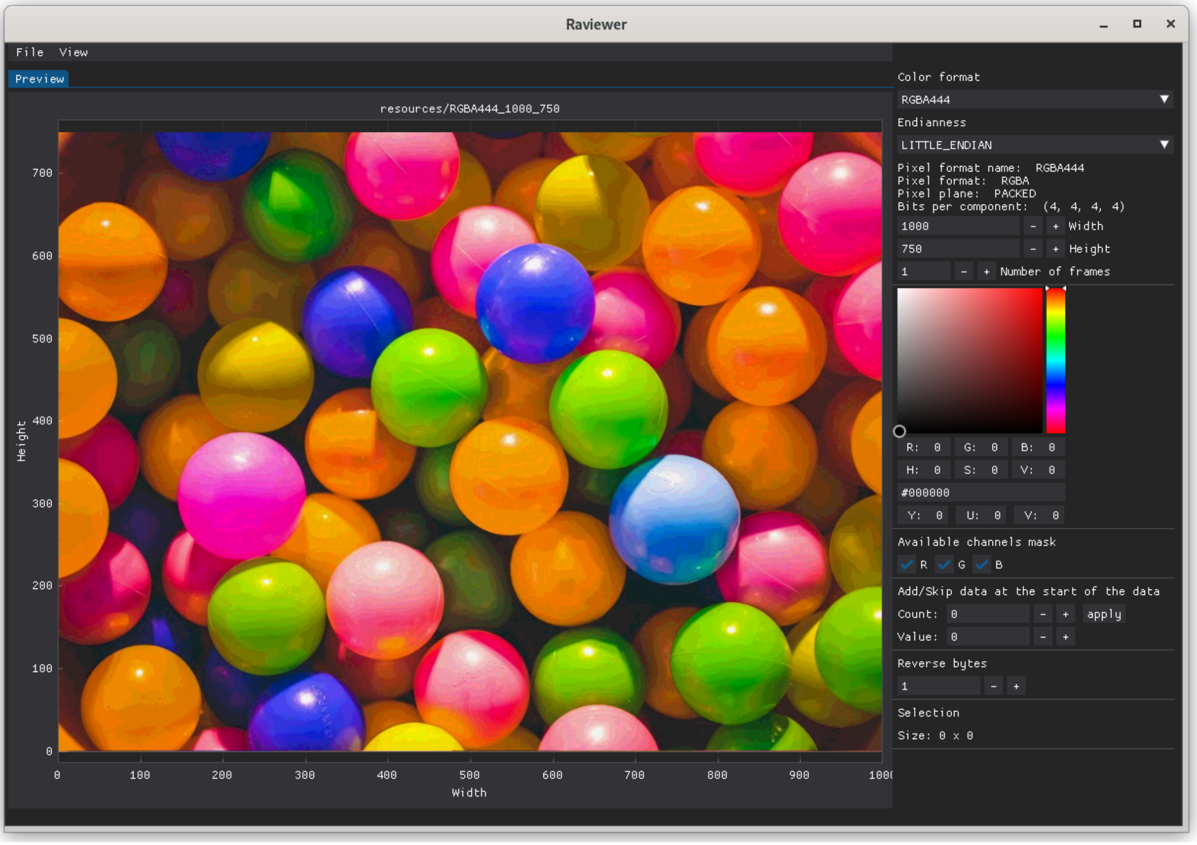
There are 2 types of endianness to choose from in the Raviewer's options: Big Endian and Little Endian. When using Big Endian, the most significant byte is placed at the byte with the lowest address, while in Little Endian, the lowest address is occupied by the least significant byte.

To accommodate that, you may need to change the endianness setting in the right side menu:

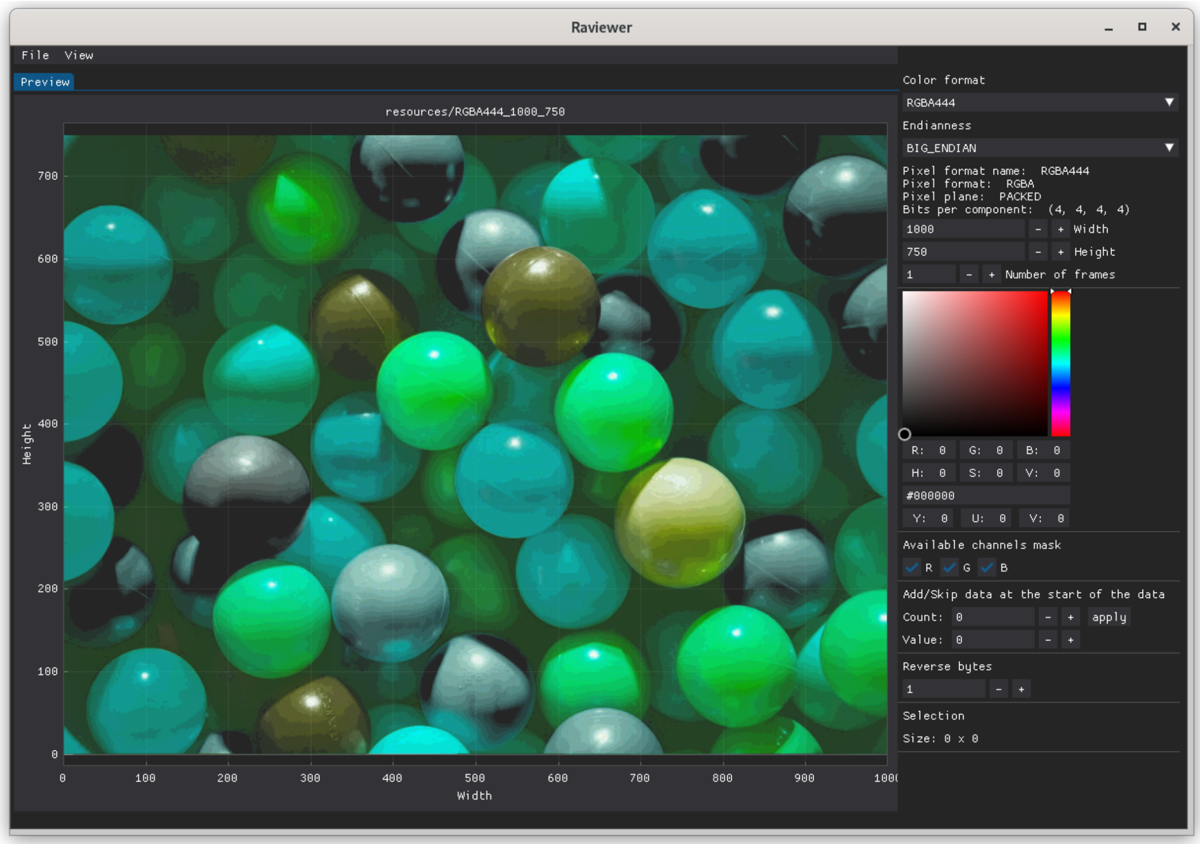


Each color format stores the data differently, and selecting the right endianness is necessary for the frame to render correctly. A pixel in RGBA444 format consists of 2 bytes. The first bytes contain information about the B and A channels, and the second one about R and G.

Take a look at this RGBA444 frame:



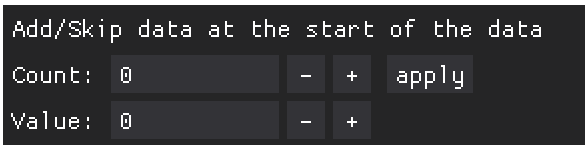
If we change the endianness, we swap the order of the 2 bytes constituting the color format, so the B channel swaps with R and the A channel swaps with G, making the picture look like this:



4.3 Adding and skipping bytes

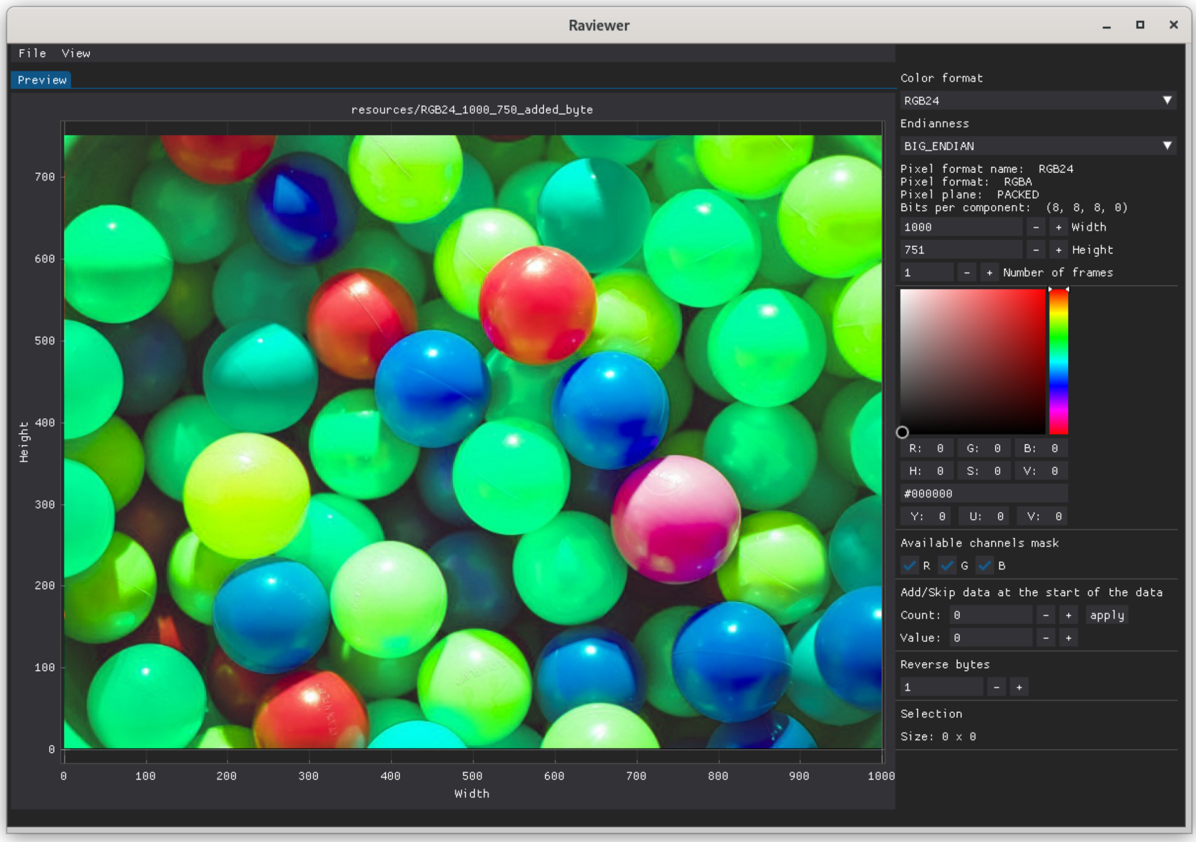
Some RAW files may include additional data before the actual picture, like header information. You may need to remove or replace the first bytes to display the image correctly. The data can be skipped or replaced with a value of your choosing.

You can add or skip data at the beginning of your binary data by changing the **Count** and the **Value**:

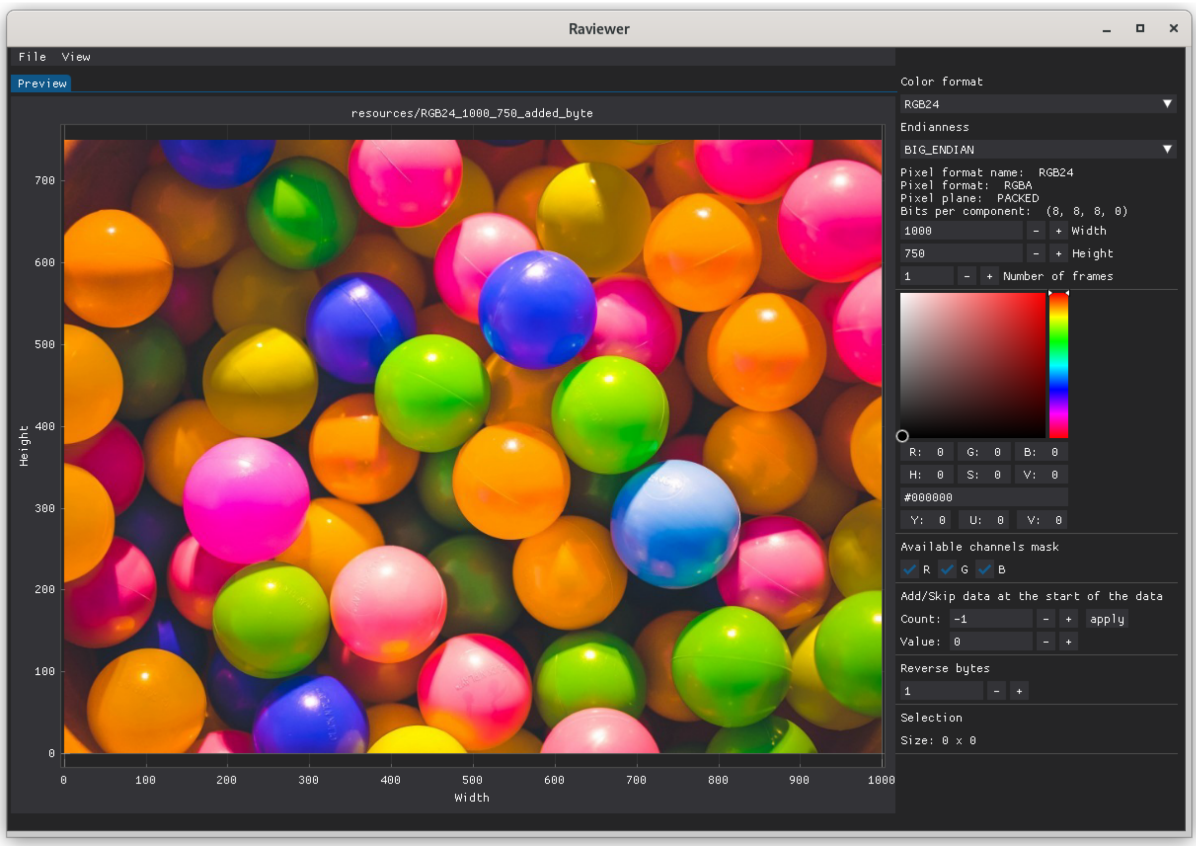


Changing the **Count** changes the number of bytes to append (or skip), while **Value** changes the value of bytes to append.

Some frames, like RGB24 shown below, have an additional byte in front of the actual picture data, which prevents Raviewer from displaying it correctly:



After skipping it (by setting the count to -1), the frame displays as it should:

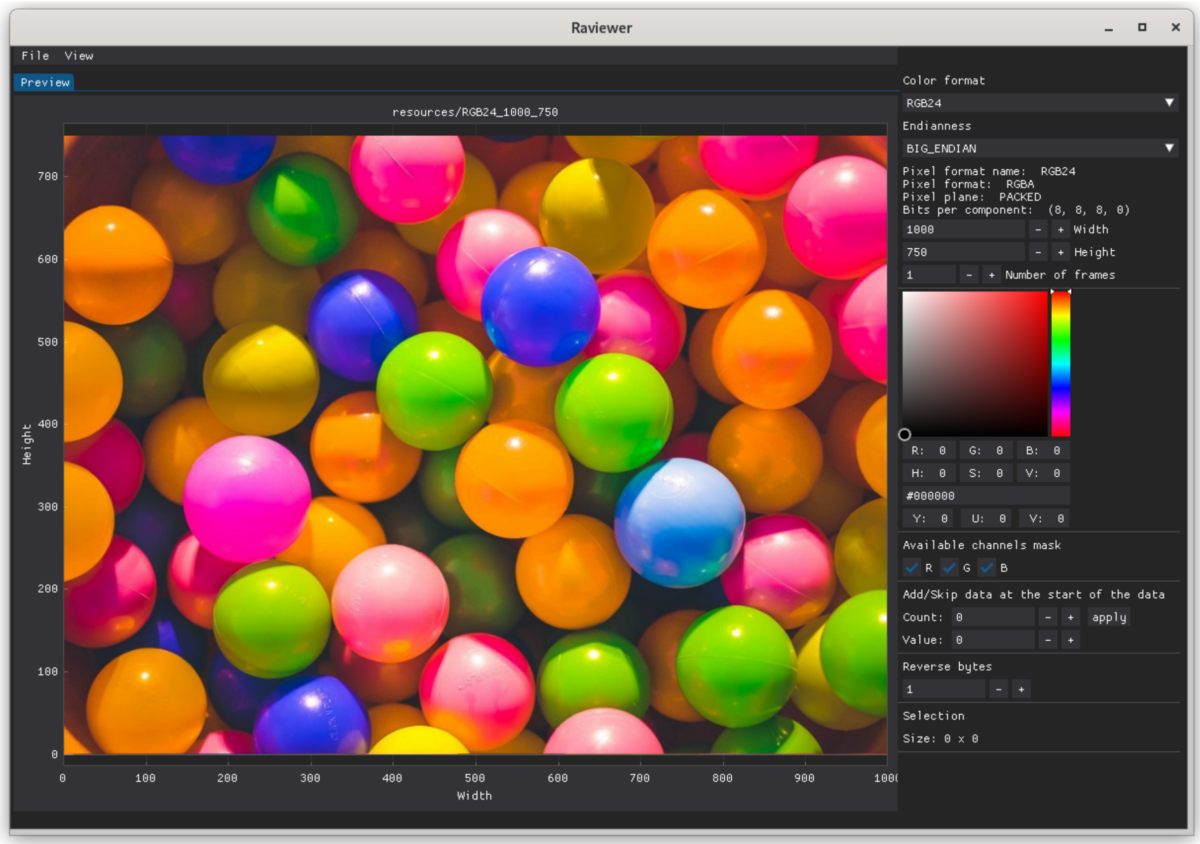


4.4 Reversing bytes

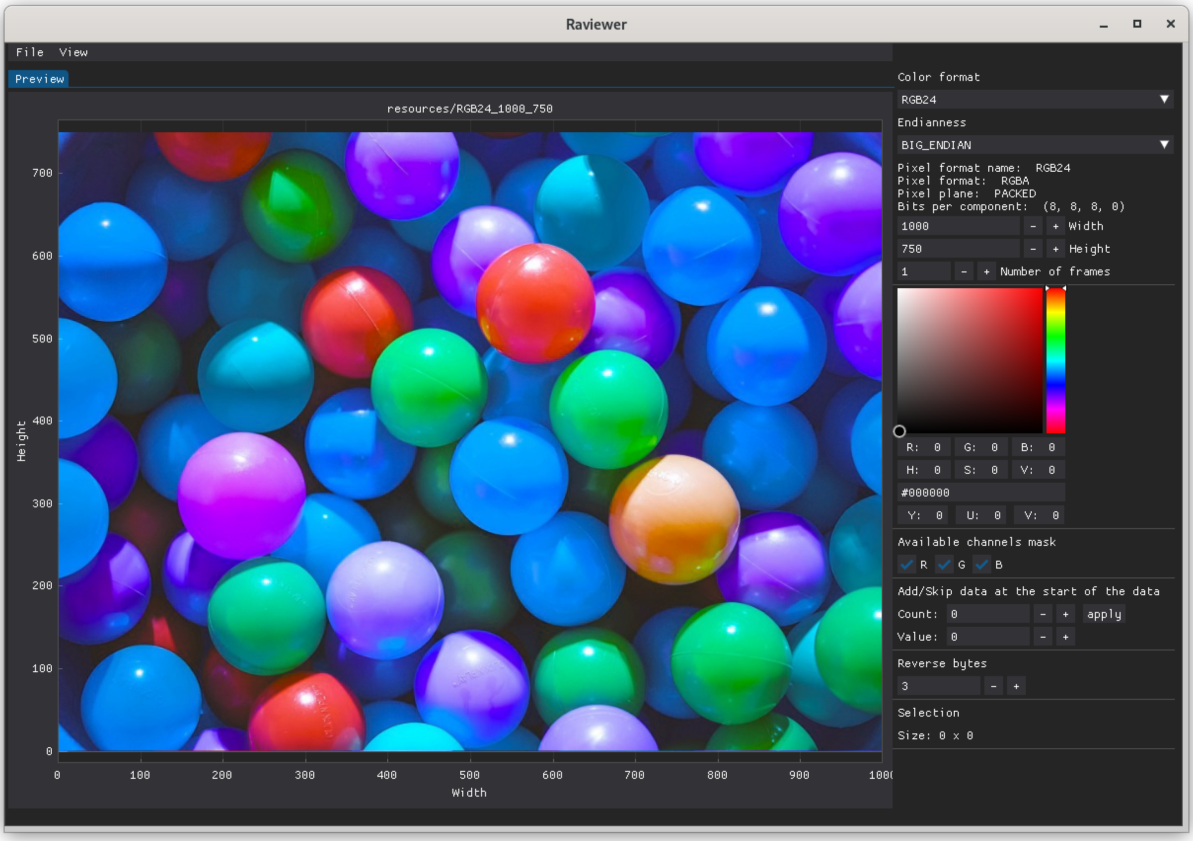
In some cases, the picture may not look properly after the initial loading of your RAW data. One of the reasons for this might be the reversed order of bytes in the imported file.

You can reverse the order of the bytes in your input data using the right-side menu. To do it, change the value in the **Reverse bytes** setting (the default value is 1).

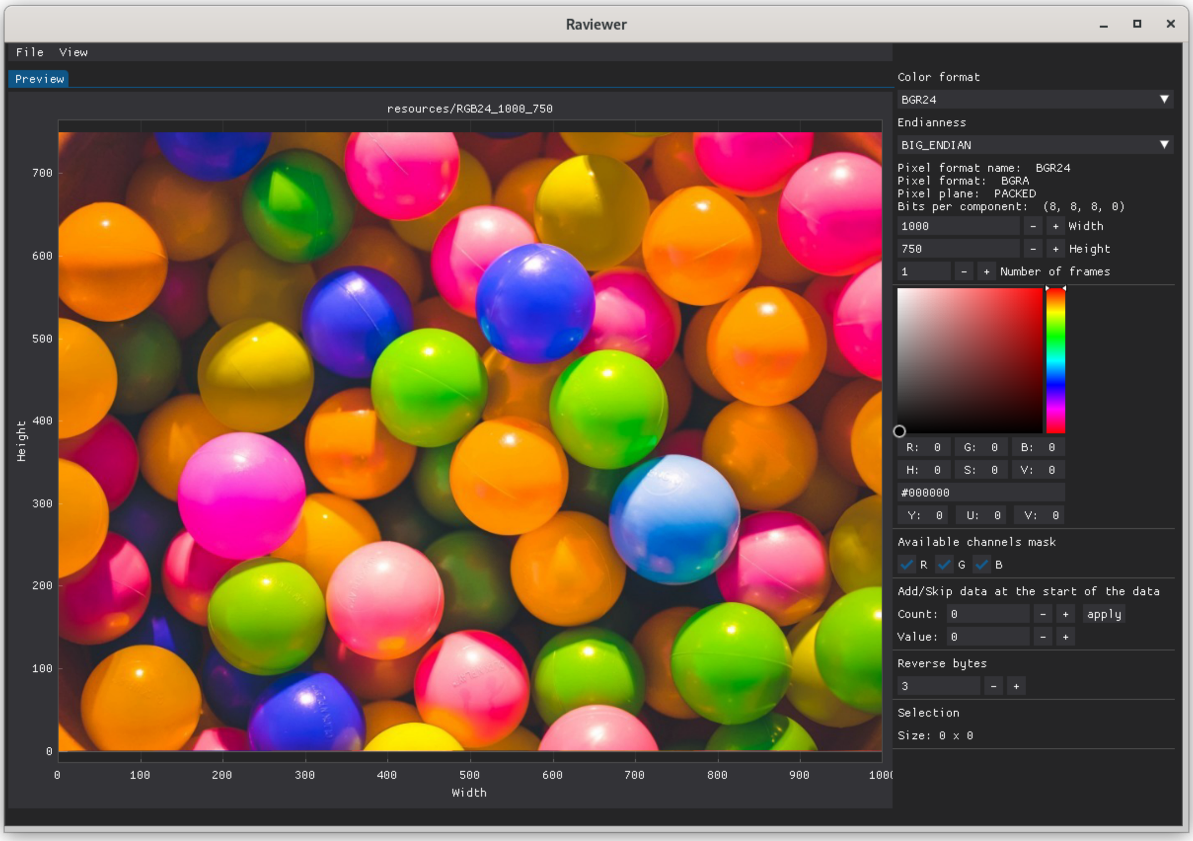
Reversing bytes enables you to expose similarities between some color formats. By loading the RGB24 frame:



Then set the value of the reverse byte setting to 3:



Your frame will now be compatible with the BGR24, and after changing the color format setting to it, it will display correctly:



4.5 Hexadecimal preview mode

Using hexadecimal preview mode, you can inspect the raw data of your picture by analyzing its hex values instead of inspecting its pixels. It helps you determine whether the picture has been converted or modified, so you can use the proper tools to analyze it.

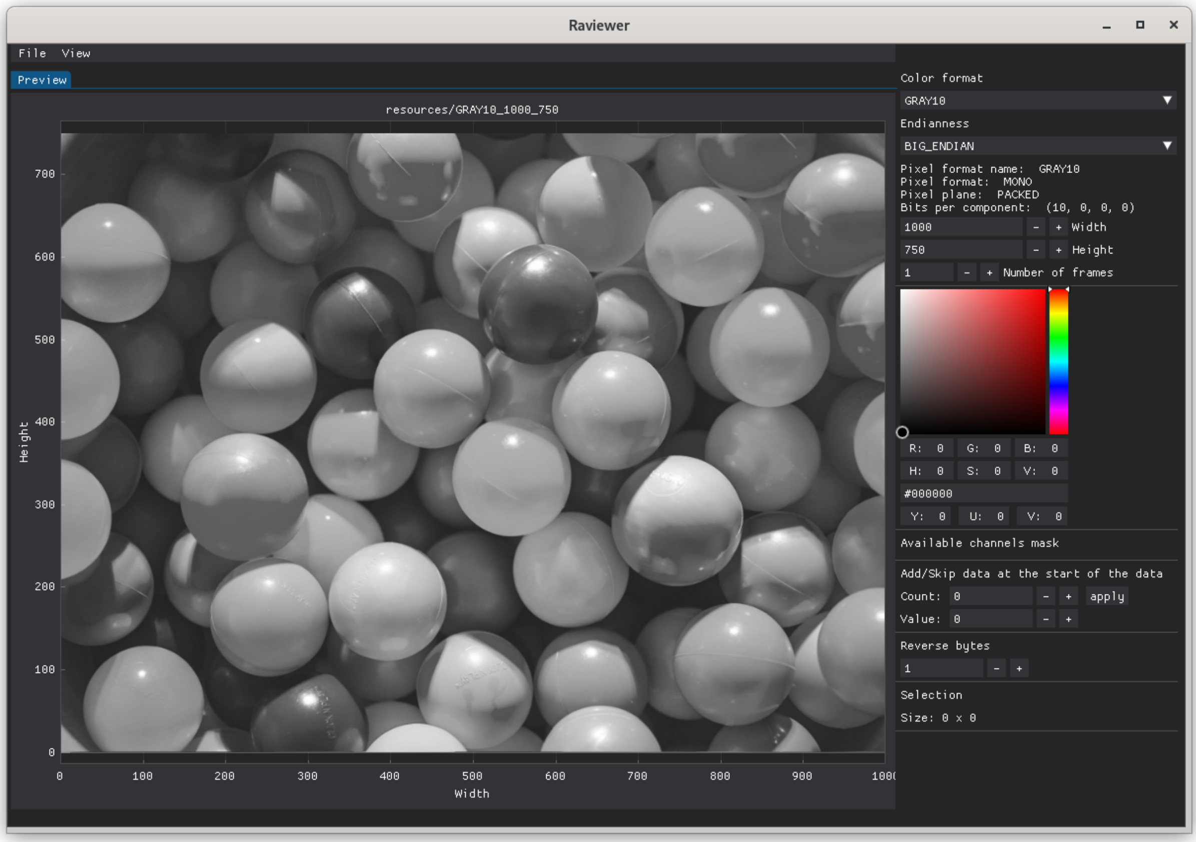
This tool gives you the ability to, for example, see whether the pixel has added padding, which is not possible to determine by just looking at it.

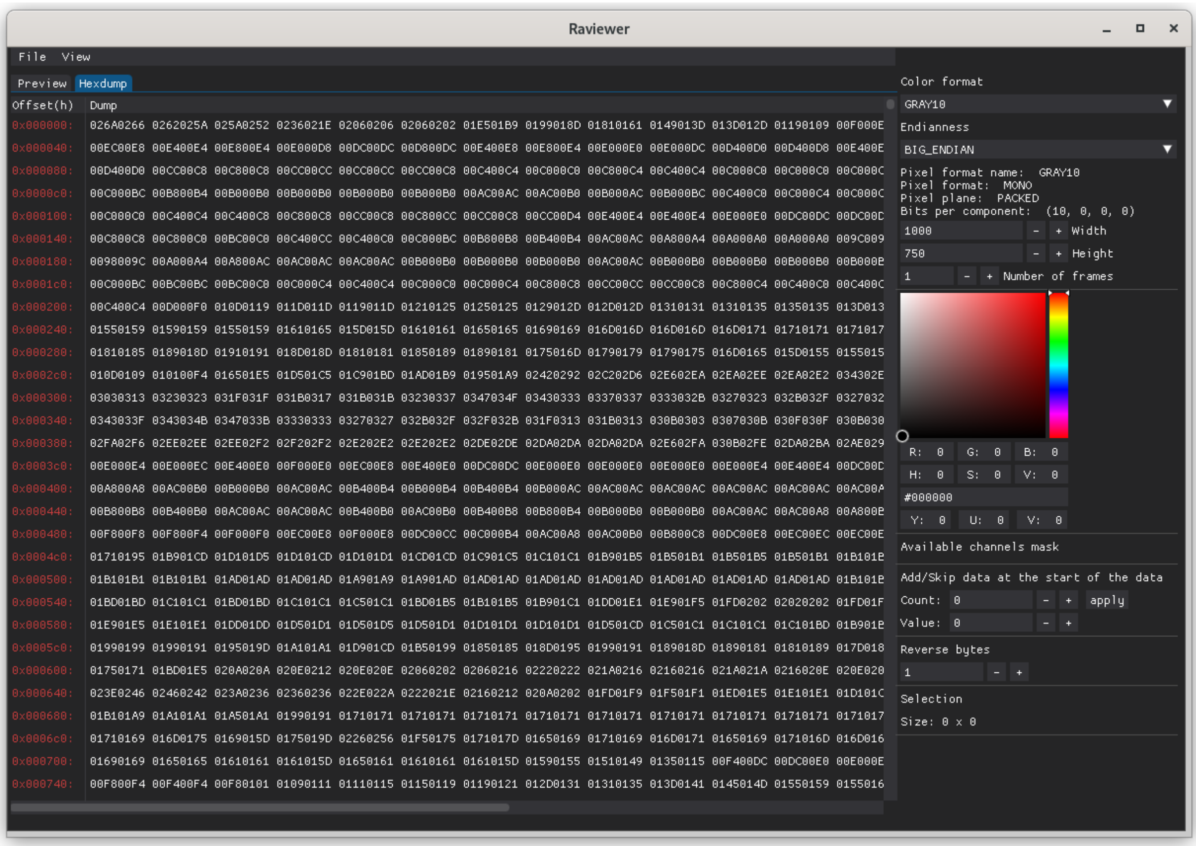
To inspect the hex dump of the imported file, you need to click **View** > **Hexdump** to turn on this view. It will now be visible next to the **Preview** view:

Offset(h)	Dump
0x000000:	FF7EFB7B F579E96B DA65D564 C454AC4A 9941833C 763B6633 582C532B 522C4F2D 542D522D 532D512A 512A502A 552D562C 4F2C4F2
0x000040:	482B4629 4529462A 452A4529 43274226 42284128 40273F28 3F283F27 3E263D26 3F273C25 3A243824 38243725 37243725 3524372
0x000080:	24222523 22262326 23252226 23242426 31253026 2E262B26 2A262527 25252226 23252323 20232226 21231F23 1B241923 11240F2
0x0000c0:	04240725 0C250E25 10241224 12241224 11231224 14241424 15241824 17241B26 1D251E24 20242622 29222A20 2A212D22 3022302
0x000100:	31284629 613A6440 623E6841 69406444 6B426C43 6C446E44 72467448 7549774A 7C4D7E4C 7D4C8050 814D824E 834F854F 8552865
0x000140:	93579757 99589857 95539755 9853934D 9350984D 904A814B 7F4B873D 7D3A6E3B 742E6F2D B855F83B ED3BEA34 E82AF472 F892FD9
0x000180:	F5AAFFAD FFABFFA8 FFAAFDB3 FFBCFBB7 FFB3FFB0 FEAEFDB2 FBB2F8B2 FABEFFBE FFB8FFBB FFB6FFB3 FEAFFFB1 FFAFFFA9 FFA7FEA
0x0001c0:	FE9EFC9B FC9CFF9C FC98FE97 FE95FD94 F996FF9D FF9DFF8A F87EFA7F FB69D004 67236D22 67236327 69256325 57296225 6025632
0x000200:	2F242C27 272A232A 252C272A 2E273023 26282528 2329222A 222A2328 26292829 292A2828 25272527 28282828 2B292C28 2E242C2
0x000240:	6F297028 71246E23 6A255A23 49213D20 3E224C23 62256E23 6E226C24 6C227532 9655AA66 B36BB469 B367AF67 AA68A56A A366A26
0x000280:	9F659F65 9D659D65 9C649C65 9D659C66 9D659D65 9E649E64 A164A164 A264A264 A568A868 A866A967 AA67A865 A466A46A 9A7C9D8
0x0002c0:	A679A478 A2769F74 9F76A074 A073A073 A0709C6D 9A6E9A6D 986B9669 96679462 8D669160 9760A45B B961B550 AE48B14C B44CAF4
0x000300:	A546D954 FB55FA57 FE52FA4B F54EFF4F FE4AFC4B FB4CFC48 FB44F944 F949FD53 F85AFB58 FC52FE51 FF4BFF46 FE41FB3D FD3AFC3
0x000340:	FF14FF12 FF11FF0C FA03FB02 FD01FE00 FF00FF00 FF00FE01 FE01FE01 FD00FD00 FA01FA02 FC00ED17 FF54FC18 F60BFA00 FF00FF0
0x000380:	FE00FE00 FD00FE00 FF00FD00 FC00FA00 EC03E000 8C106822 59296425 65266627 6C296B29 6D2C742F 7A2F7E2E 832E8A32 9031953
0x0003c0:	C645CA47 D24CD851 E04FE74E EF4BEF45 E341DC3B D63CD53D D342D43A CA35C730 BD33BA32 BA33BB37 7D107B13 771A6E1A 631E602
0x000400:	2C362B36 2C372C34 2D352D33 2D332B2E 2B2F2B30 2D302D2E 2B302C2F 292C282D 2C302C30 2B2E292C 2B2C2A2B 2A2D2A2D 292C292
0x000440:	272D272D 272D272F 272C252D 242C252E 242D262D 242B242C 25322436 24562076 24782485 1F922185 228F2390 25912693 2695259
0x000480:	28A1269F 279E269D 25992597 23952497 23932392 248F248E 258C248A 23892489 23882588 25882587 238C248E 24902591 2492249
0x0004c0:	22912390 248C248C 2690248E 238F218E 228D208A 2089228A 228F2388 21842283 23842382 217E1F79 2960226B 38314018 3D23401
0x000500:	451A461A 481B491A 4B1A4B18 4B194F1A 4D174E18 4F174F18 51165117 51165213 5312560F 590E5B0B 5C0D5A0E 570B530A 530E531
0x000540:	48183F19 3A1F3B1D 2E212E1D 42314027 3C303244 285C6297 8EA499A8 9CAA9EA9 9FCB9FB2 A3B39BB8 A5B3AAB9 AABAA8B6 A9BAACE
0x000580:	ADBAB2BE B8C3BBC0 B4BDB4BF B4BCAEBB ACB9ACB9 AAB7A8B5 A6B5A3B2 A2B3A3B4 A0B1A0B1 A1B29FB0 9DAC9AAA 99AA99A8 98A897A
0x0005c0:	8699819A 75924D3C 1E272025 24282729 25282626 29252526 252B2329 232A232A 262C262D 262C252E 263A263A 293C2A3C 2B3E284
0x000600:	2B442946 2847294A 2A482947 28482947 28452744 28432942 25462642 27392830 272E282E 252C212B 29252822 24232423 2726272
0x000640:	25292328 24212628 44145D0D 67006800 6707670B 68046A00 68006600 66006600 64006501 65016501 65006500 65006500 6400650
0x000680:	64016400 64006500 68026802 66016700 66006A00 6B006900 70007400 79047D08 7C027C04 7E057E04 79007801 77047702 7500730
0x0006c0:	6F026A00 66036306 67006000 630D6818 53084A09 48114C15 4D0F4A11 49134915 4B124A12 47134517 48094B17 5422542E 50344B4
0x000700:	45784380 48895094 5C935A95 56945193 4A944694 42923E8F 3A8B3689 30892A8B 268C228D 1F891786 0F880B85 09850682 0283018
0x000740:	01840083 00830082 03820086 007E097B 2B92478C 0B72057C 00820082 0080017A 007C017A 007A0078 00730071 006F006A 0067006

Now you can inspect the contents of your binary files in hexadecimal and ASCII form.

In the Hexdump tab, we can see that in GRAY10 format, every pixel has padding that consists of 0s.


Raviewer
File View
Preview
resources/GRAY10_1000_750
Height
Width
Color format
GRAY10
Endianness
BIG_ENDIAN
Pixel format name: GRAY10
Pixel format: MONO
Pixel plane: PACKED
Bits per component: (10, 0, 0, 0)
Width
Height
Number of frames
Available channels mask
Add/Skip data at the start of the data
Count:
Value:
apply
Reverse bytes
Selection
Size: 0 x 0


Raviewer
File View
Preview Hexdump
Offset(h) Dump
Color format
GRAY10
Endianness
BIG_ENDIAN
Pixel format name: GRAY10
Pixel format: MONO
Pixel plane: PACKED
Bits per component: (10, 0, 0, 0)
Width
Height
Number of frames
Available channels mask
Add/Skip data at the start of the data
Count:
Value:
apply
Reverse bytes
Selection
Size: 0 x 0

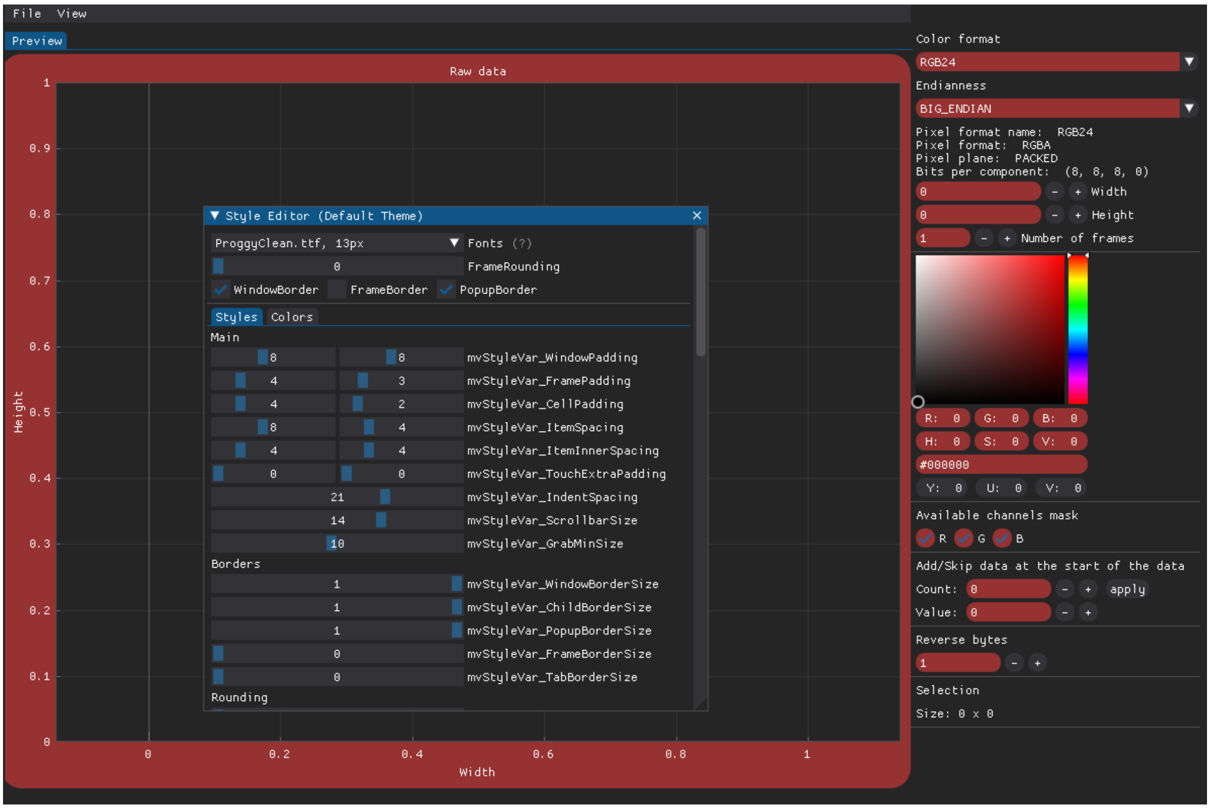
4.6 Theme manager

Raviewer uses Dear PyGui to create its user interface. It allows you to easily change various aspects of your GUI, like colors, fonts, and many more. ou can find more detailed information on managing the themes in the appropriate section of the Dear PyGui documentation.

To create a new theme, you must edit the `theme_config.py` file in the `raviewer/styles_config`. To edit the global theme, simply add the formulas in the appropriate line:

```
with dpg.theme() as items.theme.general:
    with dpg.theme_component(dpg.mvAll):
        pass
        dpg.add_theme_color(dpg.mvThemeCol_FrameBg, (150, 50, 50), category=dpg.mvThemeCat_Core)
        dpg.add_theme_style(dpg.mvStyleVar_FrameRounding, 20, category=dpg.mvThemeCat_Core)
        dpg.show_style_editor()
```

The last three lines modify the background color to red, make the window frames more round, and open the style editor when you launch Raviewer:



You can find a complete list of modifiable elements here.

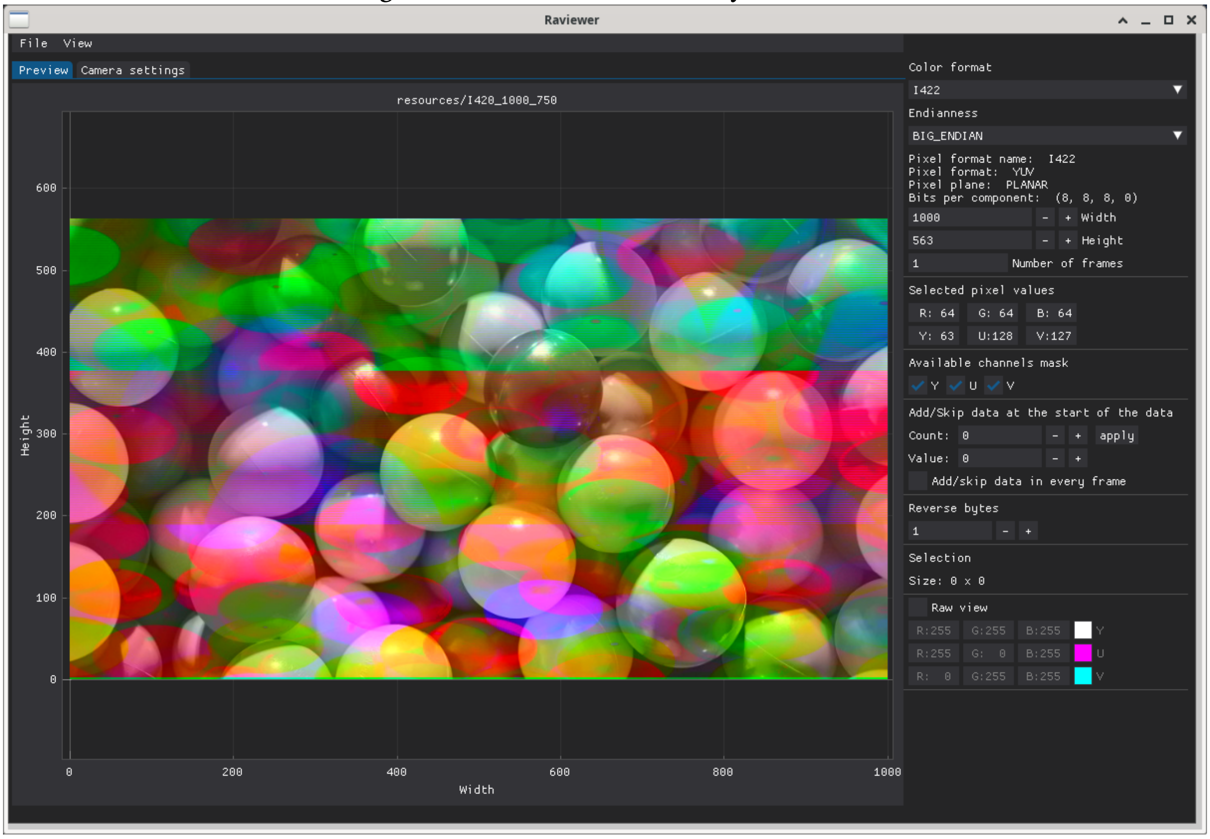
4.7 Raw view

Raw view is used for displaying images before conversion. The displayed image is rendered by assigning a channel to every pixel and giving it an appropriate color.

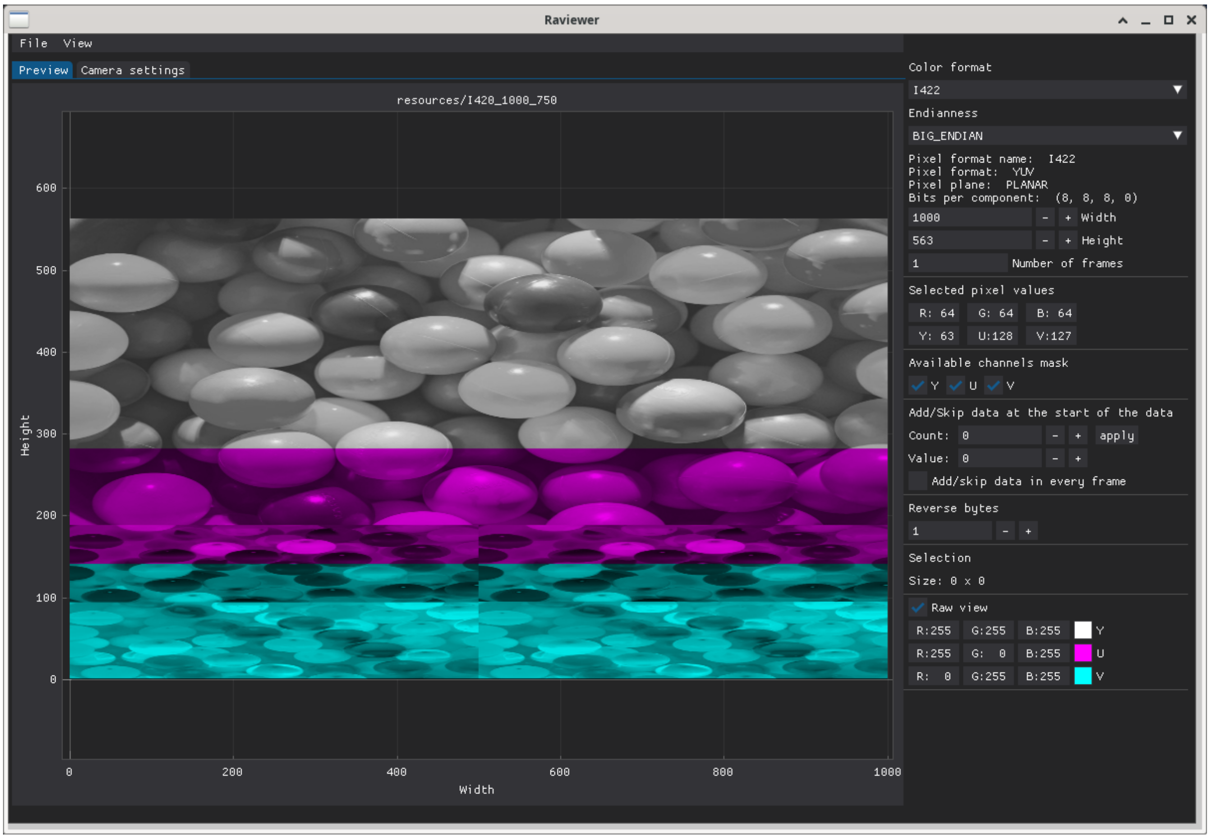
To enable raw view, check **Raw view** checkbox and select colors for presented channels.

Raw view simplify the process of understanding the image format delivered by the device. Therefore, it can be used for determining the correct color format.

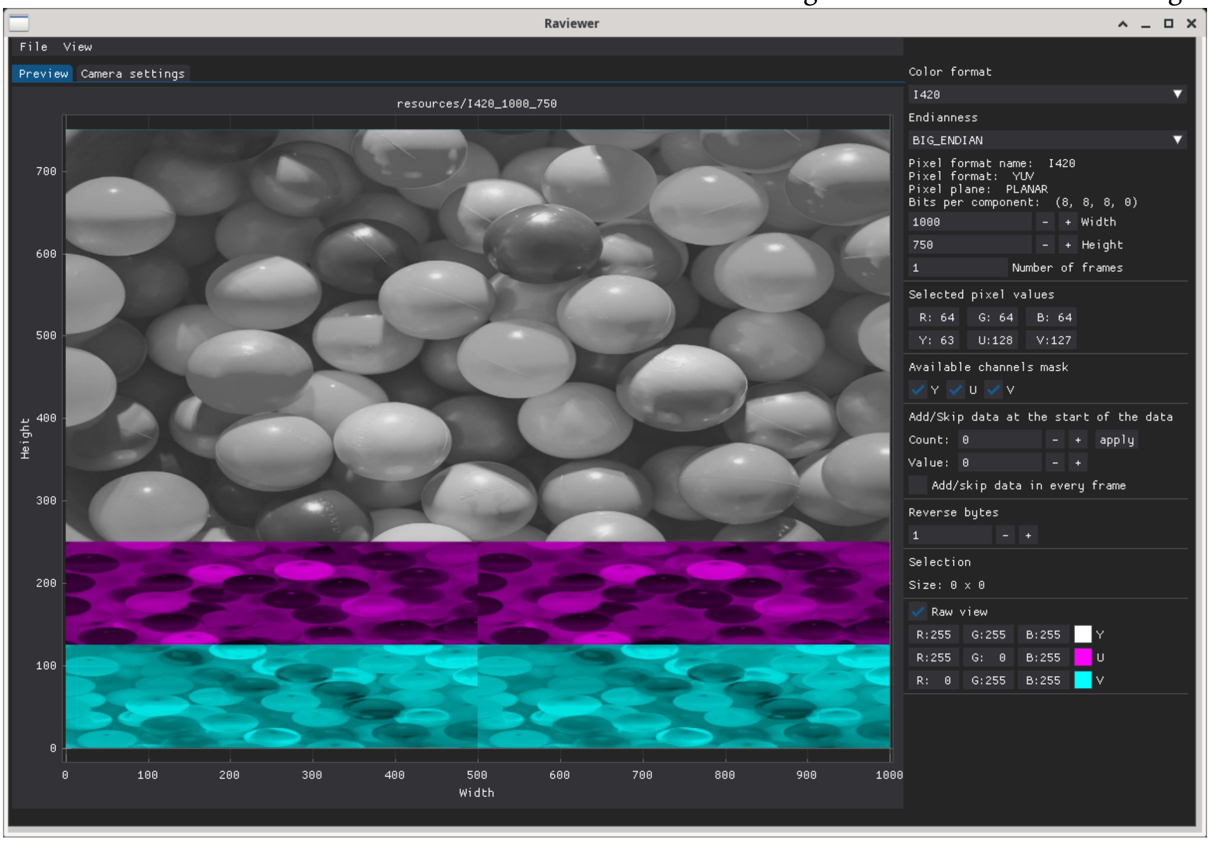
Here is an image with incorrectly chosen color format:



Raw view shows that channels are misaligned with the image:



The correct color format's channels are aligned with the image:



CHAPTER

FIVE

SUPPORTED COLOR FORMATS

5.1 RGB

The RGB color model uses 3 colors: **R**ed, **G**reen, and **B**lue to construct all the colors. Each parameter in this format represents the intensity of the colors, expressed on a scale dependent on its bit depth.

Table 5.1: RGB pixel formats

Name	VL42 Identifier
RGB332	V4L2_PIX_FMT_RGB332
ARGB444	V4L2_PIX_FMT_ARGB444
RGBA444	V4L2_PIX_FMT_RGBA444
ABGR444	V4L2_PIX_FMT_ABGR444
BGRA444	V4L2_PIX_FMT_BGRA444
ARGB555	V4L2_PIX_FMT_ARGB555
RGBA555	V4L2_PIX_FMT_RGBA555
ABGR555	V4L2_PIX_FMT_ABGR555
BGRA555	V4L2_PIX_FMT_BGRA555
RGB565	V4L2_PIX_FMT_RGB565
BGR24	V4L2_PIX_FMT_BGR24
RGB24	V4L2_PIX_FMT_RGB24
ABGR32	V4L2_PIX_FMT_ABGR32
BGRA32	V4L2_PIX_FMT_BGRA32
RGBA32	V4L2_PIX_FMT_RGBA32
ARGB32	V4L2_PIX_FMT_ARGB32

5.2 YUV

The YUV color model consists of 3 elements:

- Y is the brightness or luminescence information
- U is the red color (chroma) difference value
- V is the blue color (chroma) difference value

Both color difference values can be calculated by subtracting the Y value from the RGB color space's blue component (for U) or red component (for V). Raviewer uses the following formulas to calculate each of the YUV values:

Y = R * .299000 + G * .587000 + B * .114000

U = R * -.168736 + G * -.331264 + B * .500000 + 128

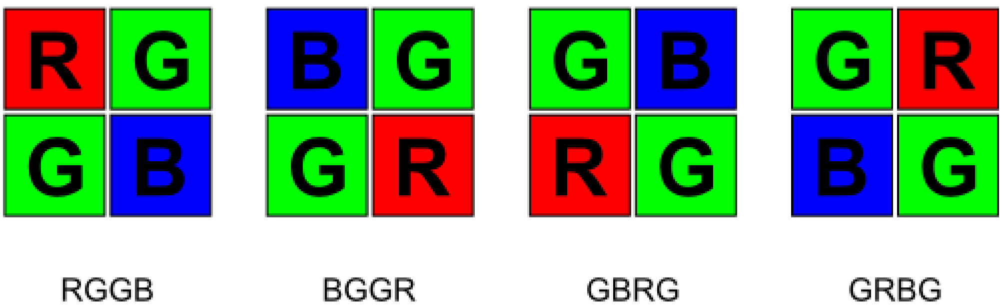
V = R * .500000 + G * -.418688 + B * -.081312 + 128

Table 5.2: YUV pixel formats

Name	VL42 Identifier	Pixel plane
UYVY	V4L2_PIX_FMT_UYVY	PACKED
YUYV	V4L2_PIX_FMT_YUYV	PACKED
VYUY	V4L2_PIX_FMT_VYUY	PACKED
YVYU	V4L2_PIX_FMT_YVYU	PACKED
NV12	V4L2_PIX_FMT_NV12	SEMI-PLANAR
NV21	V4L2_PIX_FMT_NV21	SEMI-PLANAR
I420	V4L2_PIX_FMT_YUV420	PLANAR
YV12	V4L2_PIX_FMT_YVU420	PLANAR
I422	V4L2_PIX_FMT_YUV422P	PLANAR

5.3 Bayer RGB

Bayer format is a raw video format produced by image sensors that include a Bayer filter. A Bayer filter is a color filter array in which RGB color filters are arranged on a grid of square photosensors. A Bayer filter uses two green filter elements for each red and blue filter element. The filter array can be arranged in 4 distinct patterns. Their names are derived from the order of the filters in a single 2x2 pixel square:



Bayer format is a popular raw image format used in many modern color image sensors.

Table 5.3: Bayer RGB pixel formats

Name	VL42 Identifier
RGGB	V4L2_PIX_FMT_SRGGB8
RG10	V4L2_PIX_FMT_SRGGB10
RG12	V4L2_PIX_FMT_SRGGB12
RG16	V4L2_PIX_FMT_SRGGB16

5.4 Grayscale

In the grayscale color format, each pixel only conveys intensity information. This information can be expressed on a scale dependent on its bit depth, where the minimum value represents white, and the maximum value represents black.

Table 5.4: Grayscale pixel formats

Name	VL42 Identifier
GRAY	V4L2_PIX_FMT_GRAY
GRAY10	V4L2_PIX_FMT_Y10
GRAY12	V4L2_PIX_FMT_Y12

5.5 Adding new color formats

Currently, two classes can be used to describe color formats: `ColorFormat` and `SubsampledColorFormat` (found in `app/image/color_format.py`). To create a new color format:

1. In `color_format.py`, add a new instance of one of the color format classes with the appropriate fields filled in under the AVAILABLE_FORMATS list.
2. Add parsing and displaying functions to the `AbstractParser` in `common.py`.You can also use other parsers from the folder or implement a new one.
3. The utility provides a proper parser by checking color format parameters (mainly `PixelFormat`) so make sure that your new color format has a valid translation of parameters to one of the parsers (you can find this functionality in `app/parser/factory.py`).

Note: Keep in mind that if you choose to implement a new parser, remember that `parse()` should return a one-dimensional `ndarray` with values read from the binary file, while `display()` should return an RGB-formatted 3-dimensional `ndarray` consisting of original color format values converted to RGB24.