

# @prism project

«AUTHOR»

«DATE-N-VERSION»

The @prism project<sup>1</sup> provides small size color palettes that can be used to create expressive color maps for graphics in different contexts.

---

## Last changes

---

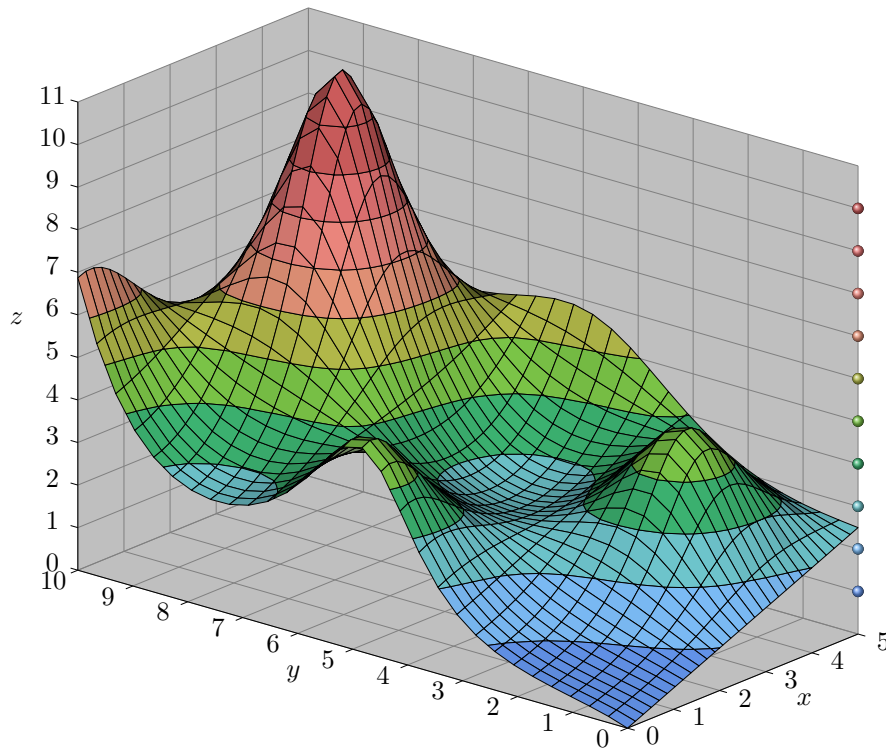
<sup>1</sup>The name comes from “*@ · esthetic P · roducts for R · epresenting I · nformative S · cientific M · aps*”. This name is a double play on words: [1] a prism is where light is split into an informative spectrum, symbolizing how data or visuals are decomposed into meaningful color and style, and [2] where light meets the prism, it breaks down into an informative spectrum (“@” can be read “at”).

# Contents

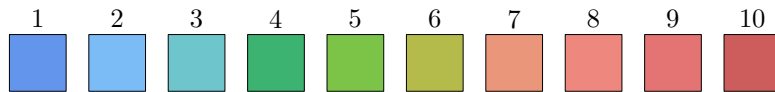
<b>I. Motivations</b>	<b>3</b>
<b>II. Where do the color palettes come from?</b>	<b>4</b>
<b>III. Reuse from...</b>	<b>4</b>
1. Matplotlib	4
2. Asymptote and Scientific Colour Maps	4
<b>IV. How to choose a palette?</b>	<b>4</b>
<b>V. Supported implementations</b>	<b>4</b>
1. JSON, the versatile default format	4
2. luadraw palettes	5
a. Description	5
b. Use a luadraw palette	5
<b>VI. Contribute via Git</b>	<b>6</b>
1. Complete the translations	6
a. The <code>en</code> folder	6
b. The <code>changes</code> folder	6
c. The <code>status</code> folder	6
d. The <code>README.md</code> and <code>LICENCE.txt</code> files	6
e. New translations	7
2. Improving the source code	7
<b>VII. History</b>	<b>7</b>

## I. Motivations

Originally, this project was born out of a desire to enhance `luadraw` with a set of color palettes to easily produce something like the following 3D plot.



Technically, a finite list of colors is provided to `luadraw` which then uses linear interpolation to calculate the intermediate colors. In the previous case, the finite color palette used is defined as follows.

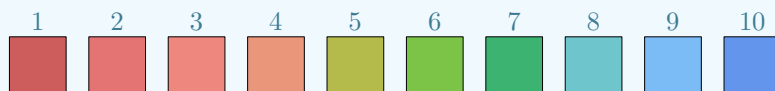


Using this palette, `luadraw` is able to produce the following spectrum, allowing us to create the graph above.

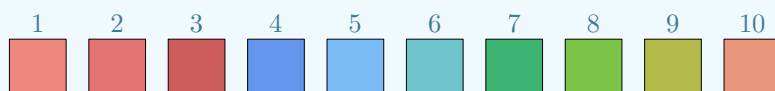


### **Note.**

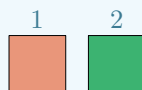
Using the `luadraw` implementation of `@prism`, see the section 2, we can display the palettes below made from the previous one named `'GeoRainbow'`. Each `luadraw` instruction used is given below each palette.



```
getPal('GeoRainbow', {reverse = true})
```



```
getPal('GeoRainbow', {shift = 3})
```



```
getPal('GeoRainbow', {extract = {7, 15, 4}})
```

## II. Where do the color palettes come from?

Most of color palettes are obtained from [Matplotlib](#) and [Scientific Coulour Maps](#) by segmenting their color maps into 15 values.<sup>2</sup> We don't keep all the palettes in accordance with the following rules.

- **No repetition.** Some [Matplotlib](#) palettes are repeated.<sup>3</sup> In this case, we keep the first one regarding to the lexicographical order.
- **No reversed version.** Contrary to [Matplotlib](#),<sup>4</sup> [@prism](#) never offers the reversed version of a palette as a fixed data.

In addition to the [Matplotlib](#) and [Scientific Coulour Maps](#) palettes, there are [@prism](#) creations.

**Note.**

*If you are interested in adding palettes, go to the section 2.*

## III. Reuse from...

### 1. [Matplotlib](#)

Here are the important points to remember when using a palette similar to what [Matplotlib](#) offers.

1. [@prism](#) uses a standardized notation in camel case format. Therefore, [Matplotlib](#) palette names such as `berlin` and `gist_heat` become `Berlin` and `GistHeat` respectively.
2. All names ending with the suffix `_r` correspond to a reversal of the palette color order. These palettes are not integrated into the [@prism](#) project. However, the implementations provide ways to easily obtain these reversed palettes, as well as sub-palettes, and also palettes obtained by shifting the colors.

### 2. [Asymptote](#) and [Scientific Coulour Maps](#)

It is sufficient to apply the naming standardization explained in the previous section: see point 1.

## IV. How to choose a palette?

The complete set of 128 palettes is visible in use cases in the following documents.

1. `showcase-en-std.pdf` is a document using a colored theme on a white background.
2. `showcase-en-dark.pdf` is a document using a colored theme on a black background.

## V. Supported implementations

The implementations are inside the folder `products`.

### 1. JSON, the versatile default format

By default, a file `palettes.json` is provided to allow unsupported coding languages to also integrate [@prism](#) palettes. Here are the first line of this file.

```
{
  "Accent": [
    [0.498039, 0.788235, 0.498039],
    [0.690196, 0.705881, 0.757298],
    [0.882352, 0.721568, 0.661437],
    [0.99477, 0.835294, 0.550326],
    [0.913289, 0.935947, 0.610021],
    [0.306317, 0.487581, 0.680174],
    [0.700653, 0.146404, 0.562091],
```

<sup>2</sup>[Asymptote](#) is also used, but to date, [Asymptote](#) offers nothing more than [Matplotlib](#), despite different implementations.

<sup>3</sup>Surely for historical reasons.

<sup>4</sup>Most of the [Matplotlib](#) color maps have a reversed version named by adding the suffix `_r`. Perhaps this is for performance reasons...

```

    [0.855772, 0.162962, 0.316775],
    [0.671459, 0.366448, 0.159041],
    [0.4, 0.4, 0.4]
  ],
  ...
}

```

## 2. luadraw palettes

### a. Description

You can use `@prism` palettes with `luadraw` which is a package that greatly facilitates the creation of high-quality 2D and 3D plots via `Lua` and `TikZ`.

#### Note.

*Initially, the `@prism` project was created to provide ready-to-use palettes for `luadraw`.*

### b. Use a luadraw palette

The `Lua` palette names all use the prefix `pal` followed by the name available in the file `palettes.json`. You can access a palette by two ways.

- `palGistHeat` is a `Lua` variable.
- `getPal('GistHeat')` and `getPal('palGistHeat')` are equal to `palGistHeat`.

#### Note.

*The `Lua` palette variables are arrays of arrays of three floats. Here is the definition of `palGistHeat`.*

```

palGistHeat = {
  {0.0, 0.0, 0.0},
  {0.105882, 0.0, 0.0},
  {0.211764, 0.0, 0.0},
  {0.317647, 0.0, 0.0},
  {0.429411, 0.0, 0.0},
  {0.535294, 0.0, 0.0},
  {0.641176, 0.0, 0.0},
  {0.752941, 0.003921, 0.0},
  {0.858823, 0.145098, 0.0},
  {0.964705, 0.286274, 0.0},
  {1.0, 0.42745, 0.0},
  {1.0, 0.57647, 0.152941},
  {1.0, 0.717647, 0.435294},
  {1.0, 0.858823, 0.717647},
  {1.0, 1.0, 1.0}
}

```

There are also some options. To explain how this works, let's consider the following use case.

```

mypal = getPal(
  'GistHeat',
  {
    extract = {2, 5, 8, 9},
    shift   = 1,
    reverse = true
  }
)

```

To simplify the explanations, we will refer to the colors in the standard palette 'GistHeat' as `coul_1`, `coul_2`, etc. The options are then processed in the following order.

1. `{coul_2, coul_5, coul_8, coul_9}` is the result of the extraction.

2. {coul\_9, coul\_2, coul\_5, coul\_8} comes from the shifting applied to the extracted palette (colors move to the right if `shift` is positive).
3. {coul\_8, coul\_5, coul\_2, coul\_9} is the reversed version of the shifted palette.

## VI. Contribute via Git

### Caution.

*Never use the `main` branch, which is used to freeze the latest stable versions of projects in the mono repository <https://github.com/projetmbc/for-writing>.*

### 1. Complete the translations

#### Important.

*Although we're going to explain how to translate the documentation, it doesn't seem relevant to do so, as English should suffice these days.*

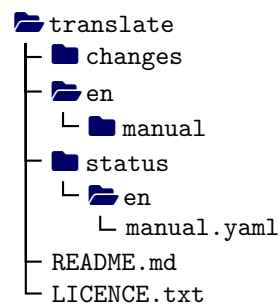


Figure 1: Simplified view of the translation folder

The translations are roughly organized as in figure 1 where just the important folders for the translations have been “opened”.<sup>5</sup> A little further down, the section e explains how to add new translations.

#### a. The `en` folder

This folder, managed by the author of @prism, contains files easy to translate even if you're not a coder.

#### b. The `changes` folder

This folder is a communication tool where important changes are indicated without dwelling on minor modifications specific to one or more translations.

#### c. The `status` folder

This folder is used to keep track of translations from the project's point of view. Everything is done via well-commented YAML files, readable by a non-coder.

#### d. The `README.md` and `LICENCE.txt` files

The `LICENCE.txt` file is aptly named, while the `README.md` file takes up in English the important points of what is said in this section about new translations.

<sup>5</sup>This was the organization on October 26, 2025.

## e. New translations

### Note.

*The folder `manual` is reserved for documentation. It contains TEX files that can be compiled directly for real-time validation of translations.*

### Warning.

*Only start from the `en` folder, as it's the responsibility of the `@prism` author.*

*Let's say you want to add support for Italian from files written in English.*<sup>6</sup> To do this, you must use `Git` as follows.

1. Via <https://github.com/projetmbc/for-writing/tree/aprism/@prism>, recover the entire project folder. Do not use the `main` branch, which is used to freeze the latest stable versions of projects in the single <https://github.com/projetmbc/for-writing> repository,.
2. In the `@prism/contrib/translate` folder, create an `it` copy of the `en` folder, with the short name of the language documented in the page “*IETF language tag*” from Wikipedia.
3. Once the translation is complete in the `it` folder, share it via <https://github.com/projetmbc/for-writing/tree/aprism/@prism> using a classic `git push`.

## 2. Improving the source code

Participation as a coder is made via the repository <https://github.com/projetmbc/for-writing/tree/aprism/@prism> corresponding to the `@prism` development branch. Here is what you can do, details can be found in the file <https://github.com/projetmbc/for-writing/blob/aprism/@prism/contrib/products/README.md>.

1. Create new palettes in one of the existent implementations.
2. Propose a new implementation.

## VII. History

---

<sup>6</sup>As mentioned above, there is no real need for the `doc` folder.