Graphics::Toolkit::Color

Herbert „Lichtkind“ Breunung

What is G::T::C for ?

- find a set of colors

Goals of G::T::C

- find a set of colors
  - for screen related purposes (Graphics)
    - no optics, print, cloths etc.

Goals of G::T::C

- find a set of colors
  - for screen related purposes (Graphics)
  - minimum knowledge required (Toolkit)
    - accept any usual format
    - converts implicitly
    - accepts names

Goals of G::T::C

- find a set of colors
  - for screen related purposes (Graphics)
  - minimum knowledge required (Toolkit)
  - fast / least amount of code
    - DWIM methods
    - precision underneath

Goals of G::T::C

- find a set of colors
  - for screen related purposes (Graphics)
  - minimum knowledge required (Toolkit)
  - fast / least amount of code
  - dependency free (Carp, Exporter, v5.12)
    - optionally Graphics::ColorNames::*

Goals of G::T::C

- find a set of colors
  - for screen related purposes (Graphics)
  - minimum knowledge required (Toolkit)
  - fast / least amount of code
  - dependency free (Carp, Exporter, v5.12)
    - all in Bundle::Graphics::ColorNames

Goals of G::T::C

- find a set of colors

What is G::T::C for ?

- find a set of colors
  - similar yet distinguishable
  - max different, yet harmonious

What is G::T::C for ?

- find a set of colors
  - similar yet distinguishable
  - max different, yet harmonious
  - along definable parameters

What is G::T::C for ?

- find a set of colors
  - similar yet distinguishable
  - max different, yet harmonious
  - along definable parameters (scriptable)

What is G::T::C for ?

- find a set of colors
  - similar yet distinguishable
  - max different, yet harmonious
  - along definable parameters (scriptable)
  - with useful defaults

What is G::T::C for ?

- find a set of colors
  - similar yet distinguishable
  - max different, yet harmonious
  - along definable parameters (scriptable)
  - with useful defaults
  - measure results

Goals of G::T::C

- find a set of colors
  - (none linear) ranges
    - from red to green
  - bowl (similar yet distinguishable)
    - shades of teal
  - self defined pattern

Color Name Dictionaries

Color::Library	Robert Krimen 2011
Graphics::ColorNames	R.Rothenberg 2019
Color::Rgb	Sherzod Ruzmetov 2002

- many many more

(not all) Alternatives:

Graphics::ColorObject	Alex Izvorski	2005
Color::Similarity	Mattia Barbon	2007
Graphics::ColorUtils	Philipp K. Janert	2007
Color::Fade	Noah Petherbridge	2008
Color::Calc	Claus Färber	2014
Convert::Color	Paul Evans	2023
Color::Scheme	Ricardo Signes	2023

(not all) Alternatives:

• Graphics::ColorObject	converter only, XYZ
• Color::Similarity	compute & convert, HCL
• Graphics::ColorUtils	good, but
• Color::Fade	very limited (::In)
• Color::Calc	values, no palette
• Convert::Color	values, no palette, GTC
• Color::Scheme	good, but

Alternatives:

- Graphics::ColorUtils Philipp K. Janert 2007
  - ++
    - supports RGB, YIQ, CMY, HSV, HLS
    - CSS, SVG, X11 names
  - --
    - explicit conversion (formats & names)
    - special gradients only (no complementary)

Alternatives:

- Color::Scheme Ricardo Signes 2023
  - ++
    - good OO API, RGB output
    - variation names (pastell, soft, pale)
  - --
    - RGB only, triade() => compl.(3),
    - fixed scheme size, no gradient

Alternatives:

- Best (for specialized schemes):

Color::Scheme	Ricardo Signes	2023

- Worst (can animate GIF):

Color	FigAnim 0.1	2004

G::T::C Constructor API

```
use Graphics::Toolkit::Color;


my $c = Graphics::Toolkit::Color->new( .. );
```

Same Constructor

```
use Graphics::Toolkit::Color qw/color/;


my $c = Graphics::Toolkit::Color->new( .. );
my $color_object = color( .. );
```

G::T::C Constructor API

```
use Graphics::Toolkit::Color qw/color/


my $blue = color( ‘blue‘ );
say color( [255, 0, 0] )->name; # blue
```

G::T::C Constructor API

```
my $blue = color( ... );
‘blue‘, ‘SVG:green‘,
#0000FF, #00F, [255, 0, 0]
{‘red‘=> 0, ‘green‘=> 0, ‘blue‘=> 255}
{‘H‘ => 240, ‘S‘ => 100, ‘L‘ => 50}
```

CMY CMYK HSV .. CIE Lab

G::T::C Getter

```
$color->name;
```

G::T::C Getter

```
$color->name;
```

rgb, rgb_hex, rgb_hash, hsl ....

G::T::C Getter

```
$color->name;
rgb, hsl ....
values( 'rgb' ); # list of three values
```

G::T::C Getter

```
$color->name;


values( 'rgb' ); # list of three values
```

G::T::C Getter

```
$color->name;


values( ’rgb’ ); # list of three values
’hash’ | ’char_hash’ | ’hex’ | ’red’
```

G::T::C Getter

```
$color->name;


values( 'rgb' ); # list of three values
'hash' | 'char_hash' | 'hex' | 'red'
values( 'CMYK' , 'char_hash');
 # { c => 1, m => 2, y => 3, k => 4}
```

G::T::C Getter

```
$color->name;


values( 'rgb' );
values( 'CMYK' , 'char_hash');


string # name | rgb hex : serialisation
```

G::T::C Measure Methods

```
$color->distance (
        to => $color2,
        in => 'HSL'
        notice_only => 'sl'
 );
# $color2 ~ object | scalar definition
```

G::T::C Adapter Methods

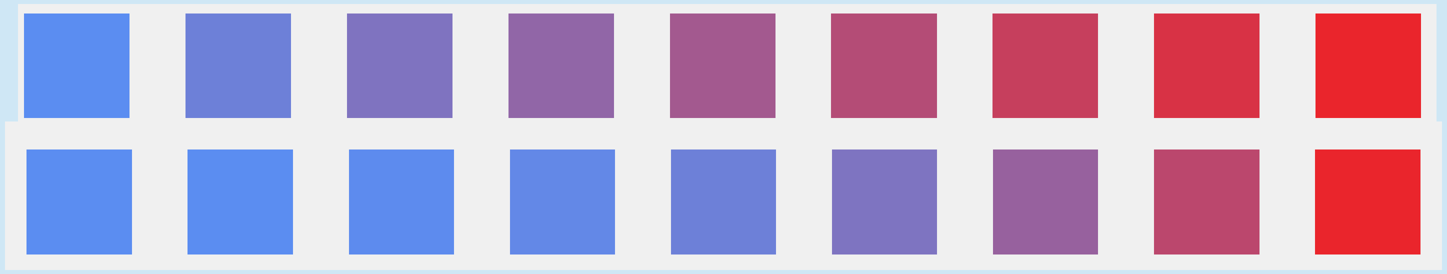
```
$color->set( Saturation => 90 );
$color->add( green => -10 );
$color->blend( with => $color2,
                                   in => 'HSL'
                                 pos => 0.4
);
```

G::T::C Methods

```
$c->gradient( to => $c2
      [, steps => 9 ]
      [, dynamic => 4 ] # -nr. slants other way
);
```

linear   :

factor 4 :



G::T::C Methods

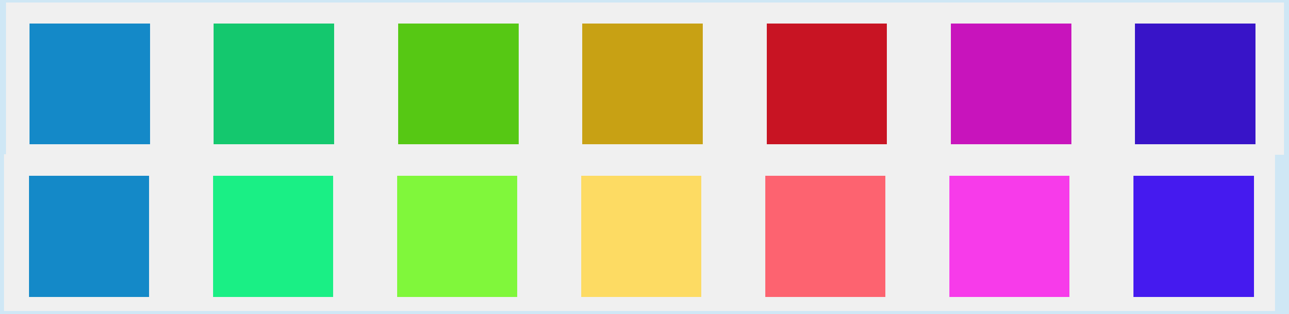
```
$c->complementary( nr [,+S] [,+L]);
```

+S ~ saturation delta (20)

+L ~ lightness delta (30)

no delta

with delta



G::T::C API

- only few methods (small surface)

G::T::C API

- only few methods (small surface)
- every method is rich (sub API)

G::T::C API

- only few methods (small surface)
- every method is rich (sub API)
- fake named arguments (speaking API)

G::T::C API

- only few methods (small surface)
- every method is rich (sub API)
- fake named arguments (speaking API)
- many names reoccur (small surface)

G::T::C API

- only few methods (small surface)
- every method is rich (sub API)
- fake named arguments (speaking API)
- many names reoccur (small surface)
- most names are optional

G::T::C Architecture

G::T::Color	- 250 LOC
G::T::Color::Constant	- 150 + 720
G::T::Color::Value	- 90 (holds:)
G::T::Color::Value::RGB	- 47 (space)
G::T::Color::Value::Space	- 110
G::T::Color::Value::SpaceBasis	- 112

G::T::C Architecture

G::T::Color	- arg handling
G::T::Color::Constant	- color names
G::T::Color::Value	- color values high lvl.
G::T::Color::Value::RGB	- special routines
G::T::Color::Value::Space	- general
G::T::Color::Value::SpaceBasis	- general

G::T::C Architecture

G::T::Color - readonly objects

G::T::Color::Constant - X11, CSS, Pantone

G::T::Color::Value - num. crunch, converter

G::T::Color::Value::RGB - special routines

G::T::Color::Value::Space - RGB, HSL,..

G::T::Color::Value::SpaceBasis - base logic

G::T::C interesting Arch.

GTC::Value - working math code for main

GTC::Value::RGB - special trim, converter

GTC::Value::Space - RGB, HSL,..

GTC::Value::SpaceBasis - base logic

G::T::C interesting Arch.

::Value	- package, holds obj.
::Value::RGB	- object, extends CODE
::Value::Space	- class, default CODE
::Value::SpaceBasis	- class, attr. of space

G::T::C interesting Arch.

::Value	- introspection, no eval
::Value::RGB	- objects, not classes
::Value::Space	- object orientation
::Value::SpaceBasis	- by composition

G::T::C interesting Arch.

Thank you