Grayness

Version 0.5.0, 2025-12-02

This Typst package provides basic image editing functions like grayscaling, inverting and cropping. Moreover, this package supports image-formats not natively available in Typst. The following formats can be used:

- BMP
- DDS
- Farbfeld
- GIF*
- HDR
- ICO
- JPEG*
- OpenEXR
- PNG*
- PNM
- QOI
- TGA
- TIFF
- WebP*
- SVG*

* Natively supported by Typst

All examples in this manual use one of the following images as their base:

WebP: Pianist Arturo Nieto Dorantes, CC BY-SA 4.0 by Laëtitia Boudaud

SVG: A traced Lamborghini Gallardo, CC BY-NC-SA 2.5 by Michael Grosberg

Available functions

- image-grayscale()
- image-show()
- image-crop()
- image-flip-horizontal()
- image-flip-vertical()
- image-blur()
- image-transparency()
- image-brighten()
- image-darken()
- image-invert()
- image-huerotate()
- image-mask()

Variables

- plg

image-grayscale

Create a grayscale-image representation of the provided imagedata (Raster or SVG)

Example:

```
#import "@preview/grayness:0.5.0": *
#let arturo = read("Arturo_Nieto-
Dorantes.webp", encoding: none)
#image-grayscale(arturo)
```

Parameters

```
image-grayscale(
  imagebytes: bytes,
  ..args
) -> content
```

imagebytes `bytes`

Raw imagedata, e.g. provided by the `read()` function

..args

extra arguments to pass to the Typst image function e.g. width, height, format, etc...

You must pass `format:"svg"` as argument if you use a SVG-image as your input.

Example:

```
#import "@preview/grayness:0.5.0": *
#let gallardo = read("gallardo.svg")
#image-grayscale(gallardo, format:"svg",
width:4cm, alt:"Lamborghini Gallardo")
```

image-show

Displays an image from bytes. This enables the usage of imageformats not natively supported by Typst

Example:

```
#import "@preview/grayness:0.5.0": *
#let arturo = read("Arturo_Nieto-
Dorantes.webp", encoding: none)
#image-show(arturo)
```

Parameters

```
image-show(
  imagebytes: bytes,
  ..args
) -> content
```

imagebytes `bytes`

Raw imagedata, e.g. provided by the `read()` function

..args

extra arguments to pass to the Typst image function e.g. width, height, format, etc...

You must pass `format:"svg"` as argument if you use a SVG-image as your input.

Example:

```
#import "@preview/grayness:0.5.0": *
#let gallardo = read("gallardo.svg")
#image-show(gallardo, format:"svg",
width:2cm, alt:"Lamborghini Gallardo")
```

image-crop

Crop the given imagedata to the specified width and height

Example:

```
#import "@preview/grayness:0.5.0": *
#let arturo = read("Arturo_Nieto-
Dorantes.webp", encoding: none)
#image-crop(
  arturo,
  crop-width: 100,
  crop-height: 120,
  crop-start-x: 190,
  crop-start-y: 95
)
```

Parameters

```
image-crop(
  imagebytes: bytes,
  crop-width: int,
  crop-height: int,
  crop-start-x: int,
  crop-start-y: int,
  ..args: arguments
) -> content
```

imagebytes `bytes`

Raw imagedata, e.g. provided by the `read()` function

crop-width `int`

Horizontal size (in pixels or “user space” in case of SVG) of the crop window. Must be >= 0.

Default: `10`

crop-height `int`

Vertical size (in pixels or “user space” in case of SVG) of the crop window. Depending on the settings of the SVG, it might preserve it’s aspect ratio even if crop-width and crop-height have a different ratio. Must be >= 0.

Default: `10`

crop-start-x `int`

Left starting coordinate (in pixels or “user space” in case of SVG) of the crop window. Depending on the settings of the SVG, it might preserve it’s aspect ratio even if crop-width and crop-height have a different ratio. Must be >= 0.

Default: `0`

crop-start-y `int`

Top starting coordinate (in pixels or “user space” in case of SVG) of the crop window. Must be >= 0.

Default: `0`

..args `arguments`

Arguments to pass to the Typst image function e.g. width, height, alt, fit, ...

You must pass `format:"svg"` as argument if you use a SVG-image as your input.

Example:

```
#import "@preview/grayness:0.5.0": *
#let gallardo = read("gallardo.svg")
#image-crop(
 gallardo,
 crop-height: 200,
 crop-width: 100,
 crop-start-x: 10,
 crop-start-y: 250,
 format: "svg"
)
```

image-flip-horizontal

Flip the provided imagedata horizontally *Example:*

```
#import "@preview/grayness:0.5.0": *
#let gallardo = read("gallardo.svg")
#image-flip-horizontal(gallardo,
format:"svg")
```

Parameters

```
image-flip-horizontal(
  imagebytes: bytes,
  ..args
) -> content
```

imagebytes `bytes`

Raw imagedata, e.g. provided by the `read()` function

..args

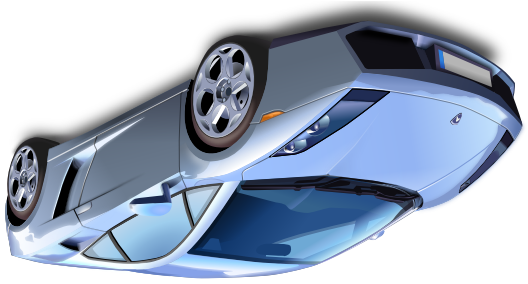
Arguments to pass to the Typst image function e.g. width, height, alt, fit, ...

You must pass `format:"svg"` as argument if you use a SVG-image as your input.

image-flip-vertical

Flip the provided imagedata vertically *Example:*

```
#import "@preview/grayness:0.5.0": *
#let gallardo = read("gallardo.svg")
#image-flip-vertical(gallardo,
format:"svg")
```



Parameters

```
image-flip-vertical(
  imagebytes,
  ..args
) -> content
```

image-blur

Performs a Gaussian blur on the imagedata.

Warning: This operation is slow, especially for large sigmas.

```
#import "@preview/grayness:0.5.0": *
##let arturo = read("Arturo_Nieto-
Dorantes.webp", encoding: none)
#image-blur(arturo, sigma:3.14)
```

Parameters

```
image-blur(
  imagebytes: bytes,
  sigma: int float,
  ..args
) -> content
```

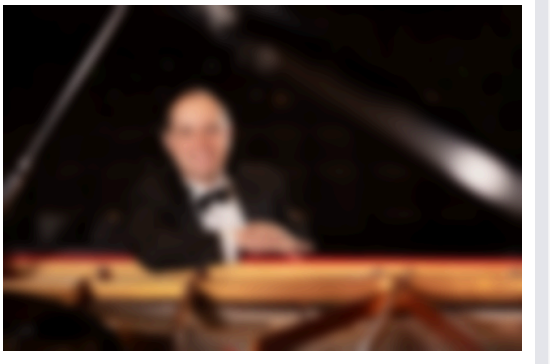
imagebytes `bytes`

Raw imagedata in any of the supported formats, e.g. provided by the `read()` function

sigma `int` or `float`

A measure of how much to blur by (standard deviation)

```
#import "@preview/grayness:0.5.0": *
#let arturo = read("Arturo_Nieto-
Dorantes.webp", encoding: none)
#image-blur(arturo, sigma:6.28)
```



Default: 5

..args

Arguments to pass to the Typst image function e.g. width, height, alt, fit, ...

You must pass `format:"svg"` as argument if you use a SVG-image as your input.

```
#import "@preview/grayness:0.5.0": *
#let gallardo = read("gallardo.svg")
#image-blur(gallardo, sigma: 40, format:
"svg")
```

image-transparency

Adds transparency to the provided image data

Example:

```
#import "@preview/grayness:0.5.0": *
//Place rectangle in background to
demonstrate transparency
#place(top+center, dx:-3cm)
[#rect(fill:blue, width:4cm, height:100%)]
//Add image over rectangle
#let arturo = read("Arturo_Nieto-
Dorantes.webp", encoding: none)
#image-transparency(arturo, alpha:50%)
```

Parameters

```
image-transparency(
  imagebytes: bytes,
  alpha: ratio,
  ..args
) -> content
```

imagebytes `bytes`

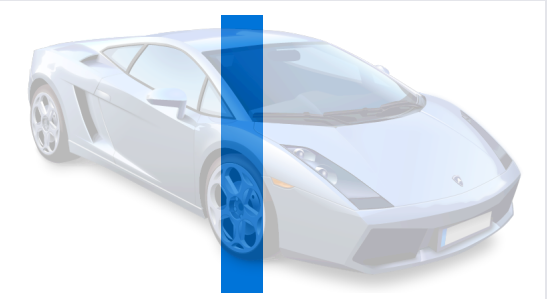
Raw imagedata in any of the supported formats, e.g. provided by the `read()` function

alpha `ratio`

Remaining amount of visibility

0% = fully transparent, 100% = fully opaque

```
#import "@preview/grayness:0.5.0": *
//Place rectangle in background to
demonstrate transparency
#place(top+center, dx:-3cm)
[#rect(fill:blue, width:4cm,
height:100%)]
#let gallardo = read("gallardo.svg")
#image-transparency(gallardo, alpha:
30%, format: "svg")
```



Default: `50%`

..args

Arguments to pass to the Typst image function e.g. width, height, alt, fit, ...

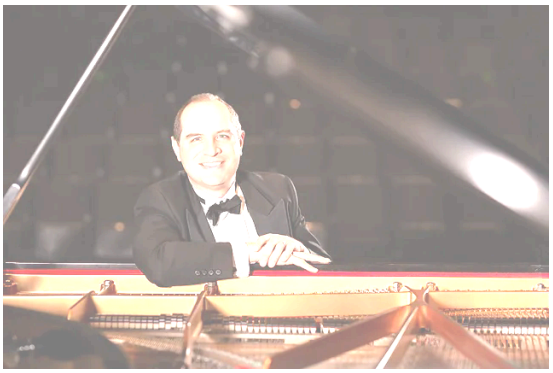
You must pass `format:"svg"` as argument if you use a SVG-image as your input.

image-brighten

Brightens the supplied image

Example:

```
#import "@preview/grayness:0.5.0": *
#let arturo = read("Arturo_Nieto-
Dorantes.webp", encoding: none)
#image-brighten(arturo, amount:50%)
```



Parameters

```
image-brighten(
  imagebytes: bytes,
  amount: ratio,
  ..args: arguments
) -> content
```

imagebytes `bytes`

Raw imagedata, e.g. provided by the `read()` function

amount `ratio`

Amount to brighten by. 0% = no brightening, 100% = completely white

Default: `50%`

..args `arguments`

Arguments to pass to the Typst image function e.g. width, height, alt, fit, ...

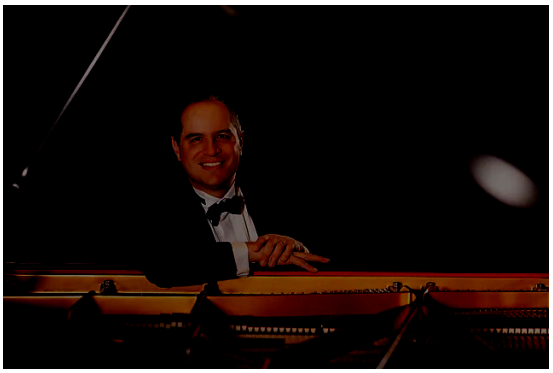
You must pass `format:"svg"` as argument if you use a SVG-image as your input.

image-darken

Darkenes the supplied image

Example:

```
#import "@preview/grayness:0.5.0": *
#let arturo = read("Arturo_Nieto-
Dorantes.webp", encoding: none)
#image-darken(arturo, amount:50%)
```



Parameters

```
image-darken(
  imagebytes: bytes,
  amount: int,
  ..args: arguments
) -> content
```

imagebytes `bytes`

Raw imagedata, e.g. provided by the `read()` function

amount `int`

Amount to darken by 0% = no darkening, 100% = completely black

Default: `50%`

..args `arguments`

Arguments to pass to the Typst image function e.g. width, height, alt, fit, ...

You must pass `format:"svg"` as argument if you use a SVG-image as your input.

```
#import "@preview/grayness:0.5.0": *
#let gallardo = read("gallardo.svg")
#image-darken(gallardo, amount:50%,
format:"svg")
```

image-invert

Inverts the colors of the supplied image

Example:

```
#import "@preview/grayness:0.5.0": *
#let arturo = read("Arturo_Nieto-
Dorantes.webp", encoding: none)v
#image-invert(arturo)
```

Parameters

```
image-invert(
  imagebytes: bytes,
  ..args: arguments
) -> content
```

imagebytes `bytes`

Raw imagedata, e.g. provided by the `read()` function

..args `arguments`

Arguments to pass to the typst image function e.g. width, height, alt, fit, ...

You must pass `format:"svg"` as argument if you use a SVG-image as your input.

```
#import "@preview/grayness:0.5.0": *
#let gallardo = read("gallardo.svg")
#image-invert(gallardo, format:"svg")
```

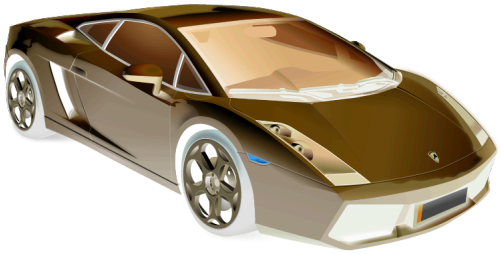


image-huerotate

Hue rotate the supplied image.

Example:

```
#import "@preview/grayness:0.5.0": *
#let arturo = read("Arturo_Nieto-
Dorantes.webp", encoding: none)
#image-huerotate(arturo, amount:100)
```

Parameters

```
image-huerotate(
  imagebytes: bytes,
  amount: int float,
  ..args: arguments
) -> content
```

imagebytes `bytes`

Raw imagedata, e.g. provided by the `read()` function

amount `int` or `float`

Number of degrees to rotate each pixels color by. 0 and 360 do nothing, the rest rotates by the given degree value. Smallest step for raster images is 1°.

Default: `0`

..args `arguments`

Arguments to pass to the Typst image function e.g. width, height, alt, fit, ...

You must pass `format:"svg"` as argument if you use a SVG-image as your input.

```
#import "@preview/grayness:0.5.0": *
#let gallardo = read("gallardo.svg")
#image-huerotate(gallardo, amount:100,
format:"svg")
```

image-mask

Update the alpha-channel of the image by applying the masking image to it. The if the mask image is not the same size as the target image, it will be resized automatically.

This function does not work with SVG data.

```
#import "@preview/grayness:0.5.0": *
#let arturo = read("Arturo_Nieto-
Dorantes.webp", encoding: none)
#let mask = read("mask.png",
encoding:none)
#image-mask(arturo, mask)
```

Parameters

```
image-mask(
  imagebytes: bytes,
  maskbytes: bytes,
  use-alpha-channel: bool,
  ..args: arguments
)
```

imagebytes `bytes`

Raw imagedata, e.g. provided by the `read()` function

maskbytes `bytes`

Raw imagedata, e.g. provided by the `read()` function

use-alpha-channel `bool`

Defines if the alpha-channel of the mask image is used (default). If set to false, the brightness of the mask image is used instead. Therefore, images without an alpha-channel can also be used as mask.

Default: `true`

..args `arguments`

Arguments to pass to the Typst image function e.g. width, height, alt, fit, ...

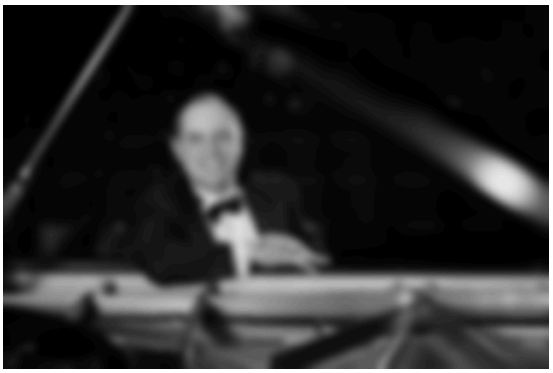
plg

Provides direct access to the webassebmly functions, which can be used to chain several operations after oneanother. The available methods are:

- `grayscale(imagebytes)`
- `svg_grayscale(image_bytes)`
- `convert(image_bytes)`
- `mask(target_image_bytes, mask_image_bytes, use_alpha)`
- `crop(image_bytes, start_x, start_y, width, height)`
- `svg_crop(image_bytes, start_x, start_y, width, height)`
- `blur(image_bytes, sigma)`
- `svg_blur(image_bytes, sigma)`
- `transparency(image_bytes, alpha)`
- `svg_transparency(image_bytes, alpha)`
- `invert(image_bytes)`
- `svg_invert(image_bytes)`
- `brighten(image_bytes, amount)`
- `svg_brighten(image_bytes, amount)`
- `huerotate(image_bytes, amount)`
- `svg_huerotate(image_bytes, amount)`

Raster Example:

```
#import "@preview/grayness:0.5.0": plg
#let arturo = read("Arturo_Nieto-
Dorantes.webp", encoding: none)
#let grayscaled-bytes =
plg.grayscale(arturo)
#let blurred-and-grayscaled =
plg.blur(grayscaled-bytes,float(5).to-
bytes(size: 4))
#image(blurred-and-grayscaled)
```



Vector Example:

```
#import "@preview/grayness:0.5.0": plg
#let gallardo = read("gallardo.svg",
encoding: none)
#let grayscaled-bytes =
plg.svg_grayscale(gallardo)
#let blurred-and-grayscaled =
plg.svg_blur(grayscaled-
bytes,float(40).to-bytes(size: 4))
#image(blurred-and-grayscaled)
```