

PyGtkImageView 1.2.0

API Documentation

April 8, 2009

Contents

Contents	1
1 Module gtkimageview	2
1.1 Classes	4
1.2 Functions	4
1.3 Variables	5
2 Class gtkimageview.AnimView	7
2.1 Methods	8
3 Class gtkimageview.IImageTool	10
3.1 Methods	10
4 Class gtkimageview.ImageNav	12
4.1 Methods	12
5 Class gtkimageview.ImageScrollWin	14
5.1 Methods	14
6 Class gtkimageview.ImageToolDragger	15
6.1 Methods	15
7 Class gtkimageview.ImageToolSelector	16
7.1 Methods	17
8 Class gtkimageview.ImageView	19
8.1 Methods	24
9 Class gtkimageview.PixbufDrawCache	33
9.1 Methods	33
10 Class gtkimageview.PixbufDrawOpts	35
10.1 Methods	35
10.2 Instance Variables	35

1 Module `gtkimageview`

This is the API reference for `PyGtkImageView` which is a set of Python bindings for the [GTK+](#) widget `GtkImageView`.

The latest releases of both these bindings and `GtkImageView` itself can always be found at <http://trac.bjournе.webfactional.com>.

`PyGtkImageView` contains a simple but full-featured image viewer widget similar to the image viewer panes in [gThumb](#) or [Eye of GNOME](#). The main class in the module is `ImageView`.

Documentation

This API documentation can be found in the `./docs` directory. HTML and PDF documentation is generated by the bash script `makedocs.sh`. To build the documentation, you need to have the following tools installed:

- [epydoc](#) 3.0 beta (install from source)
- latex (the packages `tetex-bin`, `tetex-extra` and `texlive-lang-french` in Ubuntu).
- `docutils`

Then just run the script:

```
$ ./makedocs.sh
```

The source for the documentation is found in the `./docs/gtkimageview.py` file.

Keep in mind that this documentation is **not** built from the library source. This is because while `epydoc` does support generating documentation from extension modules, the result is generally very poor ([GTK+](#) and [PyGTK](#) is partially to blame for that).

`Epydoc` has no support for extracting documentation for signals, which is why signals are here documented as methods prefixed with `sig_*`. For example, the description for the signal “`pixbuf-changed`” is found in the method description `ImageView.sig_pixbuf_changed`.

The examples should make it clear how these signals are supposed to be used.

For that reason, `epydoc` cannot automatically discern which [GTK+](#) classes `GtkImageView`’s classes subclass. Instead, see the class descriptions for inheritance information.

Release history

Major changes between version of `PyGtkImageView`. For a complete history, also see the *Release history* document in `GtkImageView`.

Major changes in 1.2.0

- `AnimView.set_anim` accepts `None`.
- `ImageView.damage_pixels` and `ImageView.image_to_widget_rect` that was supposed to be added in 1.1.0 was added for real.
- `ImageView.image_to_widget_rect` changed to ceil the width and height of the returned rectangle.
- Added example program demonstrating [progressive loading](#).

Major changes in 1.1.0

- `ImageView.damage_pixels()` and `ImageView.image_to_widget_rect()` was added.
- `IImageTool.pixbuf_changed()` got a third argument.
- All parts of the `PixbufDrawCache` API addition was added. That includes:
 - `DRAW_METHOD_CONTAINS`
 - `DRAW_METHOD_SCALE`
 - `DRAW_METHOD_SCROLL`
 - `PixbufDrawCache`
 - `PixbufDrawOpts`
- A `pygtkimageview.pc` pkg-config file is now installed with the package.
- Module-level function `library_version()` added.

Major changes in 1.0.0

None! First release. :) **Author:** [Björn Lindqvist](#)

Requires: Python 2.2+, `GtkImageView`, `PyGTK`

Version: 1.2.0

License: LGPL

Copyright: © 2007-2009 [Björn Lindqvist](#)

To Do:

- Make the epydoc generated HTML documentation a little nicer.
- Fix the PDF generation so that images can be used both in the HTML and in the PDF documentation.
- Windows installer.

1.1 Classes

- **PixbufDrawOpts**: Container class which holds options for how the pixbuf should be drawn.
(Section 10, p. 35)
- **PixbufDrawCache**: Cache that ensures fast redraws by storing the last draw operation.
(Section 9, p. 33)
- **ImageTool**: ImageTool is an interface that defines how `ImageView` interacts with objects that acts as tools.
(Section 3, p. 10)
- **ImageToolDragger**: ImageToolDragger is the default image tool for `ImageView`.
(Section 6, p. 15)
- **ImageToolSelector**: ImageToolSelector is a tool for selecting areas of an image.
(Section 7, p. 16)
- **ImageNav**: ImageNav is a popup window that shows a downscaled preview of the pixbuf that `ImageView` is showing.
(Section 4, p. 12)
- **ImageScrollWin**: Provides a widget similar in appearance to `gtk.ScrollableWindow` that is more suitable for displaying `ImageView`'s.
(Section 5, p. 14)
- **ImageView**: `ImageView` is a full-featured general purpose image viewer widget for GTK.
(Section 8, p. 19)
- **AnimView**: `AnimView` subclasses `ImageView`.
(Section 2, p. 7)

1.2 Functions

library_version()

Returns a string with the format “major.minor.micro” which denotes the runtime version of `GtkImageView` being used. Note that this is the version of the underlying C library, to retrieve the version of the Python bindings use `gtkimageview.__version__`.

```
>>> gtkimageview.library_version()
'1.5.0'
```

zooms_get_zoom_in(*zoom*)

Returns the zoom factor that is one step larger than the supplied zoom factor.

Parameters

`zoom`: a zoom factor

Return Value

a zoom factor that is one step larger than the supplied one

zooms_get_zoom_out(*zoom*)

Returns the zoom factor that is one step smaller than the supplied zoom factor.

Parameters

zoom: a zoom factor

Return Value

a zoom factor that is one step smaller than the supplied one

zooms_get_min_zoom()

Returns the minimum allowed zoom factor. **Return Value**

the minimal zoom factor

zooms_get_max_zoom()

Returns the maximum allowed zoom factor. **Return Value**

the maximal zoom factor

zooms_clamp_zoom(*zoom*)

Returns the zoom factor clamped to the minimum and maximum allowed value.

Parameters

zoom: a zoom factor

Return Value

the zoom factor clamped to the interval [min, max]

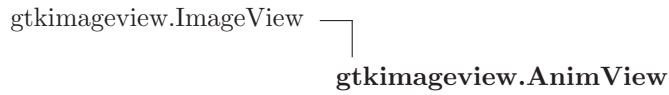
1.3 Variables

Name	Description
TRANSP_COLOR	Enumeration value to use a single color for transparent parts. Value: 1
TRANSP_GRID	Enumeration value to use a light and dark gray grid for transparent parts. Value: 2
TRANSP_BACKGROUND	Enumeration value to use the widgets background color for transparent parts. Value: 3
DRAW_METHOD_SCALE	Enumeration value to scale the area of the pixbuf to draw and put the result in cache. This is the slowest draw method as the whole area to be drawn must be rescaled. It is mostly used when no part of <code>PixbufDrawCache:s</code> cache is valid. Value: 0

continued on next page

Name	Description
DRAW_METHOD_CONTAINS	Enumeration value to get the area of the pixbuf to draw from the cache and not update the cache afterwards. Value: 1
DRAW_METHOD_SCROLL	Enumeration value to partially use the cache and scale the region not cached. The cache is updated with the result. Value: 2

2 Class `gtkimageview.AnimView`



`AnimView` subclasses `ImageView`. It has the same look and feel as its parent but is also capable of displaying GIF animations.

Keybindings

`AnimView` uses a few more keybindings in addition to those used by `ImageView`:

Key	Description
<code> GDK_ScrollUp</code>	Scrolls displaying() or resumes the animation
<code> GDK_ScrollDown</code>	Scrolls the animation one frame forward

See Also: `ImageView`, The file `./tests/ex-anim.c` program for an example of how this widget is used.

2.1 Methods

`__init__(self)`

Creates a new `AnimView` with default values. The default values are:

- `anim` : `None`
- `is playing` : `False`

Overrides: `gtkimageview.ImageView.__init__`

`step(self)`

Steps the animation one frame forward. If the animation is playing it will be stopped. Will it wrap around if the animation is at its last frame?

Read-write properties

`get_anim(self)`

Returns the current animation of the view. **Return Value**
the current animation

`set_anim(self, anim)`

Sets the pixbuf animation to play, or `None` to not play any animation.

The effect of this method is analogous to `ImageView.set_pixbuf()`. `Fitting` is set to `True` so that the whole area of the animation fits in the view. Three signals are emitted, first the `ImageView` will emit `ImageView.sig_zoom_changed` and then `ImageView.sig_pixbuf_changed`.

The default pixbuf animation is `None`. **Parameters**
`anim`: a pixbuf animation to play

See Also: `ImageView.set_pixbuf()`

`get_is_playing(self)`

Returns whether the animation is playing or not. If there is no current animation, this method will always returns `False` **Return Value**
`True` if an animation is playing, `False` otherwise.

set_is_playing(*self*, *playing*)

Sets whether the animation should play or not. If there is no current animation this method does not have any effect. **Parameters**

playing: True to play the animation, False otherwise.

Inherited from *gtkimageview.ImageView*(Section 8)

get_black_bg(), *get_fitting()*, *get_interpolation()*, *get_pixbuf()*, *get_show_cursor()*,
get_show_frame(), *get_tool()*, *get_zoom()*, *set_black_bg()*, *set_fitting()*, *set_interpolation()*,
set_pixbuf(), *set_show_cursor()*, *set_show_frame()*, *set_tool()*, *set_zoom()*

Signals

Inherited from *gtkimageview.ImageView*(Section 8)

sig_pixbuf_changed(), *sig_zoom_changed()*

Read-only properties

Inherited from *gtkimageview.ImageView*(Section 8)

get_check_colors(), *get_draw_rect()*, *get_viewport()*

Write-only properties

Inherited from *gtkimageview.ImageView*(Section 8)

set_offset(), *set_transp()*

Actions

Inherited from *gtkimageview.ImageView*(Section 8)

damage_pixels(), *image_to_widget_rect()*, *zoom_in()*, *zoom_out()*

3 Class `gtkimageview.IImageTool`

Known Subclasses: `gtkimageview.ImageToolDragger`, `gtkimageview.ImageToolSelector`

`IImageTool` is an interface that defines how `ImageView` interacts with objects that acts as tools. `ImageView` delegates many of its most important tasks (such as drawing) to its tool which carries out all the hard work. The `PyGtkImageView` package comes with two tools; `ImageToolDragger` and `ImageToolSelector`, but by implementing your own tool it is possible to extend `ImageView` to do stuff its author (thats me) didn't imagine.

`GtkImageView` uses `ImageToolDragger` by default, as that tool is the most generally useful one. However, it is trivial to make it use another tool.

```
view = gtkimageview.ImageView()
tool = gtkimageview.ImageToolSelector(view)
view.set_tool(tool)
```

Using the above code makes the view use the selector tool instead of the default dragger tool.

3.1 Methods

```
button_press(self, ev_button)
```

```
button_release(self, ev_button)
```

```
motion_notify(self, ev_motion)
```

`pixbuf_changed(self, reset_fit, rect)`

Indicate to the tool that either a part of, or the whole pixbuf that the image view shows has changed. This method is called by the view whenever its pixbuf or its tool changes. That is, when any of the following method are used:

- `ImageView.set_pixbuf()`
- `ImageView.set_tool()`
- `ImageView.damage_pixels()`

If the `reset_fit` parameter is `True`, it means that a new pixbuf has been loaded into the view. `rect` is a rectangle in image space coordinates that indicates which region of the pixbufs pixels that is modified. `rect` can be `None` which means that all image data in the pixbuf has been changed.

Parameters

`reset_fit`: whether the view is resetting its fit mode or not

`rect`: a `gtk.gdk.Rectangle` containing the changed area or `None`

See Also: `ImageView.sig_pixbuf_changed`

`paint_image(self, draw_settings, drawable)`

Called whenever the image view decides that any part of the image it shows needs to be redrawn. **Parameters**

`draw_settings`: a set of draw settings to use for this draw

`drawable`: a `gtk.gdk.Drawable` to draw the image data on

`cursor_at_point(self, x, y)`

Get the cursor to display when the mouse pointer is over the widget coordinate `(x, y)`. **Parameters**

`x`: the mouse pointers x-coordinate

`y`: the mouse pointers y-coordinate

Return Value

an appropriate `gtk.gdk.Cursor`

4 Class `gtkimageview.ImageNav`

`ImageNav` is a popup window that shows a downscaled preview of the `pixbuf` that `ImageView` is showing. The user can drag around a rectangle which indicates the current view of the image.

This class is used by `ImageScrollWin` itself. It is probably not very useful for clients of this library.

`ImageNav` has the same keybindings that `ImageView` has. All keypresses that it receives are passed along to the view.

`ImageNav` is a subclass of `gtk.Window`. **See Also:** `ImageScrollWin`

4.1 Methods

`__init__(self, view)`

Creates a new image navigator for showing thumbnails of the view. The default values are:

- `pixbuf` : `None`

`get_pixbuf(self)`

Returns the downscaled `pixbuf` of the views `pixbuf` that the image navigator shows, or `None` if that `pixbuf` has not been created yet.

The `pixbuf` is by default `None`. **Return Value**

the `pixbuf` in the image window that this navigator shows, or `None` if it has not been created.

`grab(self)`

`release(self)`

show_and_grab(*self*, *center_x*, *center_y*)

Show the navigator centered around the point (*center_x*, *center_y*) and grab mouse and keyboard events. The grab continues until a left mouse button release event is received which causes the widget to hide. **Parameters**

center_x: x coordinate of center position

center_y: y coordinate of center position

5 Class `gtkimageview.ImageScrollWin`

Provides a widget similar in appearance to `gtk.ScrollableWindow` that is more suitable for displaying `ImageView`'s.

In particular, this widget draws a crosshair icon in the bottom right corner. Pressing that icon brings up an `ImageNav` which shows a thumbnailed overview of the viewed `pixbuf`.

`ImageScrollWin` is a subclass of `gtk.Table`. **See Also:** [GtkScrolledWindow](#) the GTK widget that `ImageScrollWin` mimics, `ImageNav`

5.1 Methods

<code>__init__</code> (<i>self</i> , <i>view</i>)
Creates a new <code>ImageScrollWin</code> containing the view.
The widget is built using four subwidgets arranged inside a <code>gtk.Table</code> with two columns and two rows. Two scrollbars, one navigator button (the decorations) and one <code>ImageView</code> .
When the <code>ImageView</code> fits inside the window, the decorations are hidden.

6 Class `gtkimageview.ImageToolDragger`



`ImageToolDragger` is the default image tool for `ImageView`. Its only feature is that it can drag the image around.

6.1 Methods

<code>__init__(self, view)</code>
Creates a new dragger tool. Parameters view: an <code>ImageView</code>

Inherited from `gtkimageview.IImageTool` (Section 3)

`button_press()`, `button_release()`, `cursor_at_point()`, `motion_notify()`, `paint_image()`, `pixbuf_changed()`

7 Class `gtkimageview.ImageToolSelector`



`ImageToolSelector` is a tool for selecting areas of an image. It is useful for cropping an image, for example. The tool is an implementor of the `IImageTool` interface which means that it can be plugged into an `ImageView` object by using the `ImageView.set_tool()` method.

`ImageToolSelector` changes the default display of the `ImageView`. It darkens down the unselected region of the image which provides a nice effect and makes it clearer what part of the image that is currently selected. Unfortunately, this effect is somewhat incompatible with how `ImageNav` behaves and that widget will show the image without darkening it.

The tool also changes the default behaviour of the mouse. When an `ImageToolSelector` is set on an `ImageView`, mouse presses do not “grab” the image and you cannot scroll by dragging. Instead mouse presses and dragging is used to resize and move the selection rectangle. When the mouse drags the selection rectangle to the border of the widget, the view autoscrolls which is a convenient way for a user to position the selection.

Please note that `ImageToolSelector` draws the image in two layers. One darkened and the selection rectangle in normal luminosity. Because it uses two draw operations instead one one like `ImageToolDragger` does, it is significantly slower than that tool. Therefore, it makes sense for a user of this library to set the interpolation to `gtk.gdk.INTERP_NEAREST` when using this tool to ensure that performance is acceptable to the users of the program.

Zoom bug

There is a small bug in `ImageToolSelector` that becomes apparent when the zoom factor is greater than about 30. The edge of the selection rectangle may in that case intersect a pixel:

The bug is caused by bug [389832](#) in `gdk-pixbuf`. There is no way to solve this bug on `ImageView`’s level (but if someone knows how, I’d really like to know).

7.1 Methods

<p><code>__init__</code>(<i>self</i>, <i>view</i>)</p> <hr/> <p>Creates a new selector tool for the specified view with default values. The default values are:</p> <ul style="list-style-type: none"> • selection : (0, 0) - [0, 0] <p>Parameters</p> <p>view: an <code>ImageView</code></p>
--

Inherited from `gtkimageview.IImageTool`(Section 3)

`button_press()`, `button_release()`, `cursor_at_point()`, `motion_notify()`, `paint_image()`, `pixbuf_changed()`

Signals

<p><code>sig_selection_changed</code>(<i>cls</i>)</p> <hr/> <p>The selection-changed signal is emitted when the selection rectangle on the selector is moved or resized. It is intended to be used by applications that wants to print status information. For example:</p> <pre>def sel_changed_cb(selector): rect = selector.get_selection() if not rect.width or not rect.height): print 'No selection' else: fmt = 'The selection is %d, %d - %d, %d' print fmt % (rect.x, rect.y, rect.width, rect.height) ... selector.connect('selection-changed', sel_changed_cb)</pre>
--

Read-write properties

get_selection(*self*)

Returns the current selection rectangle in image space coordinates. If either the width or the height of the rectangle is zero, then nothing is selected and the selection should be considered inactive. **Return Value**
the selection rectangle

See Also: `sig_selection_changed` for an example

set_selection(*self*, *rect*)

Sets the selection rectangle for the tool. Setting this attribute will cause the widget to immediately repaint itself if its view is realized.

This method does nothing under the following circumstances:

- If the views pixbuf is `None`.
- If `rect` is wider or taller than the size of the pixbuf.
- If `rect` equals the current selection rectangle.

If the selection falls outside the pixbufs area, its position is moved so that it is within the pixbuf.

Calling this method causes the `sig_selection_changed` signal to be emitted.

The default selection is (0,0) - [0,0]. **Parameters**
`rect`: selection rectangle in image space coordinates

8 Class `gtkimageview.ImageView`

Known Subclasses: `gtkimageview.AnimView`

`ImageView` is a full-featured general purpose image viewer widget for GTK. It provides a scrollable, zoomable pane in which a `pixbuf` can be displayed.

Keybindings

When focused, `GtkImageView` responds to the following keybindings:

Key	Description	Function
<code>GDK_KEY_Alt</code> , <code>GDK_KEY_equal</code> , <code>GDK_KEY_plus</code>	Zoom in one step	<code>zoom_in_one_step()</code>
<code>GDK_KEY_Alt</code> , <code>GDK_KEY_minus</code>	Zoom out one step	<code>zoom_out_one_step()</code>
<code>GDK_KEY_F11</code>	Zoom to 100%	<code>zoom_to_100_percent()</code>

Key	Description	Function
GDK/Zoom	zoom to 200%	<code>GtkZoom()</code>
GDK/Zoom	zoom to 300%	<code>GtkZoom()</code>
GDK/Zoom	fitting to True	<code>GtkZoom()</code>
GDK/Space Up, GDK_Up + view GDK_SHIFT_MASK	a page upwards	<code>GtkZoom()</code>
GDK/Space Down, GDK_Down + view GDK_SHIFT_MASK	a page downwards	<code>GtkZoom()</code>
GDK/Space Left, GDK_Left + view GDK_SHIFT_MASK	a page leftwards	<code>GtkZoom()</code>

- **Image coordinates:** each coordinate represents a pixel in the image. The range of valid coordinates goes from $(0,0)$ - $(p.w,p.h)$, where $p.w$ and $p.h$ is the width and height of the image.
- **Widget coordinates:** each coordinate represents a pixel in the image view widgets coordinate system. The range of valid coordinates goes from $(0,0)$ - $(a.w,a.h)$ where $a.w$ and $a.h$ is the allocated width and height of the widget. Naturally, these coordinates are only valid for as long as the widget is realized.
- **Zoom coordinates:** this coordinate system is the most frequently used coordinate system in `ImageView`. The range of valid coordinates goes from $(0,0)$ - $z(p.w,p.h)$ where $p.w$ and $p.h$ are the width and height of the image and z is the current zoom of the view. In other words, this coordinate system is simply the image coordinate system scaled.

Settings

`ImageView` has a few settings that can be configured by users of the library. For example, when showing transparent images it may in certain cases be better to draw alpha transparent parts using the widgets background color instead of the default checkerboard:

```
view.set_transp(gtkimageview.TRANSP_BACKGROUND)
```

When the window that is showing the widget is fullscreened, other settings has to be tweaked to make the view look as good as possible:

```
view.set_show_cursor(False)
view.set_show_frame(False)
view.set_black_bg(True)
```

Naturally, you should reset these settings again when the view leaves fullscreen mode.

Updating the image data

`ImageView` aggressively caches the scaled image data. This behaviour is most often beneficial and makes the widget very fast. For a concrete example, try opening a very large image (4000x2000 pixels or so) in `ImageView`. The widget will spend some time bilinearly downsampling the image. Then try minimizing and unminimizing the window. The image will reappear immediately because the view has cached it.

However, this feature means that a client application must signal to the view when it changes the pixels on the `pixbuf` that the view shows. The right way to do that is to use `ImageView.damage_pixels()`. Code that merely tries to update the view by requesting

that it should be redrawn will not work.

```
# Do some operation on the pixbuf data here..
view.queue_draw_area(10, 10, 50, 50)    # Incorrect!
```

Example

This is the minimal code needed for using `ImageView`:

```
import gtk
import gtk.gdk
import gtkimageview

win = gtk.Window()
view = gtkimageview.ImageView()
pixbuf = gtk.gdk.pixbuf_new_from_file("tests/gnome_logo.jpg")
view.set_pixbuf(pixbuf)
win.add(view)
win.show_all()
gtk.main()
```

Note that because the example doesn't use `ImageScrollWin` many nice features aren't available.

`ImageView` is a subclass of `gtk.Widget`.

8.1 Methods

`__init__(self)`

Creates a new image view with default values. The default values are:

- `black bg` : `False`
- `fitting` : `True`
- `image tool` : an `ImageToolDragger` object
- `interpolation mode` : `gtk.gdk.INTERP_BILINEAR`
- `offset` : `(0, 0)`
- `pixbuf` : `None`
- `show cursor` : `True`
- `show frame` : `True`
- `transp` : `TRANSP_GRID`

Signals

`sig_pixbuf_changed(cls)`

The `pixbuf-changed` signal is emitted when the `pixbuf` the image view shows or its image data is changed. Listening to this signal is useful if you, for example, have a label that displays the width and height of the `pixbuf` in the view.

```
def pixbuf_cb(view, label):
    new_pb = view.get_pixbuf()
    if not new_pb:
        # Empty label if no pixbuf
        label.set_text('')
        return
    label.set_text('%d, %s' % (new_pb.get_width(), new_pb.get_height()))
    ...
label = gtk.Label('')
view.connect('pixbuf-changed', pixbuf_cb, label)
```

sig_zoom_changed(*cls*)

The zoom-changed signal is emitted when the zoom factor of the view changes. Listening to this signal is useful if, for example, you have a label that displays the zoom factor of the view. Use `get_zoom()` to retrieve the value. For example:

```
def zoom_cb(view, label):
    zoom = view.get_zoom()
    label.set_text("%d%%" % int(zoom * 100))
    ...
label = gtk.Label('100%')
view.connect('zoom-changed', zoom_cb, label)
```

Read-only properties**get_viewport**(*self*)

Returns a `gtk.gdk.Rectangle` with the current viewport. If `pixbuf` is `None`, there is no viewport and `None` is returned.

The current viewport is defined as the rectangle, in zoomspace coordinates as the area of the loaded `pixbuf` the image viewer is currently showing. **Return Value**

a `gtk.gdk.Rectangle` with the current viewport or `None`

get_draw_rect(*self*)

Returns the rectangle in the widget where the `pixbuf` is painted.

For example, if the widgets allocated size is 100, 100 and the `pixbufs` size is 50, 50 and the zoom factor is 1.0, then the `pixbuf` will be drawn centered on the widget. The rectangle is then (25,25)-[50,50]. If the widget is unallocated or the `pixbuf` is `None` then `None` is returned instead.

This method is useful when converting from widget to image or zoom space coordinates. **Return Value**

the rectangle in the widget where the `pixbuf` is drawn or `None`.

get_check_colors(*self*)

Returns a tuple with the two colors used to draw transparent parts of images with an alpha channel. Note that if the transparency setting of the view is `TRANSP_BACKGROUND` or `TRANSP_COLOR`, then both colors will be equal.

```
>>> view.get_check_colors()
(6710886, 10066329)
```

Return Value

a tuple containing two color values used to draw transparent parts.

Write-only properties**set_offset**(*self*, *x*, *y*, *invalidate=False*)

Sets the offset of where in the image the image viewer should begin displaying image data.

The offset is clamped so that it will never cause the image viewer to display pixels outside the pixbuf. Setting this attribute causes the widget to repaint itself if it is realized.

If `invalidate` is `True`, the views entire area will be invalidated instead of instantly redrawn. The view is then queued for redraw, which means that additional operations can be performed on it before it is redrawn.

The difference can sometimes be important like when you are overlaying data and get flicker or artifacts when setting the offset. If that happens, setting `invalidate` to `True` could fix the problem. See the source code for [GtkImageToolSelector](#) for an example.

Normally, `invalidate` should always be `False` because it is much faster to repaint immediately than invalidating. **Parameters**

x: x-component of the offset in zoom space coordinates
y: y-component of the offset in zoom space coordinates
invalidate: whether to invalidate the view or redraw immediately

`set_transp(self, transp, transp_color=0)`

Sets how the view should draw transparent parts of images with an alpha channel. If `transp` is `TRANSP_COLOR`, the specified `transp_color` will be used. Otherwise the `transp_color` argument is ignored. If it is `TRANSP_BACKGROUND`, the background color of the widget will be used. If it is `TRANSP_GRID`, then a grid with light and dark gray boxes will be drawn on the transparent parts.

Calling this method causes the widget to immediately repaint. It also causes the `sig_pixbuf_changed` signal to be emitted. This is done so that other widgets (such as `ImageNav`) will have a chance to render a view of the `pixbuf` with the new transparency settings.

The default values are:

- `transp` : `TRANSP_GRID`
- `transp_color` : `0x000000`

Parameters

`transp`: transparency type to use when drawing transparent images

`transp_color`: color to use when drawing transparent images.

Read-write properties

`get_pixbuf(self)`

Returns the `pixbuf` this view shows. **Return Value**
the `pixbuf` this view shows

set_pixbuf(*self*, *pixbuf*, *reset_fit*=True)

Sets the pixbuf to display, or `None` to not display any pixbuf.

Normally, `reset_fit` should be `True` which enables fitting. Which means that, initially, the whole pixbuf will be shown.

Sometimes, fitting should not be reset. For example, if `ImageView` is showing an animation, it would be bad to reset fitting for each new frame. The parameter should then be `False` which leaves the fit mode of the view untouched.

This method must also be used to signal to the view that the contents of the pixbuf it display has been changed. This is the right way to force the redraw:

```
view.set_pixbuf(view.get_pixbuf(), False)
```

If `reset_fit` is `True`, the `sig_zoom_changed` signal is emitted, otherwise not. The `sig_pixbuf_changed` signal is also emitted.

The default pixbuf is `None`. **Parameters**

`pixbuf`: the pixbuf to display
`reset_fit`: whether to set fitting or not

get_zoom(*self*)

Get the current zoom factor of the view. **Return Value**
the current zoom factor

set_zoom(*self*, *zoom*)

Sets the zoom of the view.

Fitting is always disabled after this method has run. The `sig_zoom_changed` signal is unconditionally emitted. **Parameters**

`zoom`: the new zoom factor

get_fitting(*self*)

Returns the fitting setting of the view. **Return Value**
`True` if the view is fitting the image, `False` otherwise

set_fitting(*self*, *fitting*)

Sets whether to fit or not. If **True**, then the view will adapt the zoom so that the whole pixbuf is visible.

Setting the fitting causes the widget to immediately repaint itself.

Fitting is by default **True**. **Parameters**

fitting: whether to fit the image or not

get_black_bg(*self*)

Returns whether the view renders the widget on a black background or not.

Return Value

True if a black background is used, otherwise **False**.

set_black_bg(*self*, *black_bg*)

If **True**, the view uses a black background. If **False**, the view uses the default (normally gray) background.

The default value is **False**. **Parameters**

black_bg: whether to use a black background or not

get_tool(*self*)

Returns views tool which is an instance of a `IImageTool`. **Return Value**
the currently bound tool

set_tool(*self*, *tool*)

Set the image tool to use. If the new tool is the same as the current tool, then nothing will be done. Otherwise `IImageTool.pixbuf_changed()` is called so that the tool has a chance to generate initial data for the pixbuf.

Parameters

tool: the image tool to use (must not be `None`)

get_interpolation(*self*)

Returns the current interpolation mode of the view. **Return Value**
the interpolation

set_interpolation(*self*, *interp*)

Sets the interpolation mode of how the view. `gtk.gdk.INTERP_HYPER` is the slowest, but produces the best results. `gtk.gdk.INTERP_NEAREST` is the fastest, but provides bad rendering quality. `gtk.gdk.INTERP_BILINEAR` is a good compromise.

Setting the interpolation mode causes the widget to immediately repaint itself.

The default interpolation mode is `gtk.gdk.INTERP_BILINEAR`. **Parameters**
interp: the interpolation to use. A `gtk.gdk.InterpType` object.

get_show_cursor(*self*)

Returns whether to show the mouse cursor when the mouse is over the widget or not. **Return Value**
True if the cursor is shown, otherwise **False**.

set_show_cursor(*self*, *show_cursor*)

Sets whether to show the mouse cursor when the mouse is over the widget or not. Hiding the cursor is useful when the widget is fullscreened.

The default value is **True**. **Parameters**
show_cursor: whether to show the cursor or not

get_show_frame(*self*)

Returns whether a one pixel frame is drawn around the pixbuf or not. **Return Value**
True if a frame is drawn around the pixbuf, otherwise **False**

set_show_frame(*self*, *show_frame*)

Sets whether to draw a frame around the image or not. When **True**, a one pixel wide frame is shown around the image. Setting this attribute causes the widget to immediately repaint itself.

The default value is **True**. **Parameters**

show_frame: whether to show a frame around the pixbuf or not

Actions

zoom_in(*self*)

Zoom in the view one step. Calling this method causes the widget to immediately repaint itself.

zoom_out(*self*)

Zoom in the view one step. The widget is immediately repainted.

damage_pixels(*self*, *rect*)

Mark the pixels in the rectangle as damaged. That the pixels are damaged, means that they have been modified and that the view must redraw them to ensure that the visible part of the image corresponds to the pixels in that image.

Damaging first causes the `IImageTool.pixbuf_changed()` to be called which allows the tool to, for example, update its cache of the pixbuf if it has one.

The signal `sig_pixbuf_changed` is emitted which enables interested listeners to update their view of the pixbuf. And finally is the visible part of the damaged rectangle queued for redraw. **Parameters**

rect: the `gtk.gdk.Rectangle` in image space coordinates to mark as dirty.

See Also: `set_pixbuf()`, `sig_pixbuf_changed`

image_to_widget_rect(*self*, *rect*)

Converts a rectangle in image space coordinates to widget space coordinates. If the view is not realized, or if it contains no `pixbuf`, then the conversion cannot be done and `None` is returned.

Note that this method may return a rectangle that is not visible on the widget.

The size of the returned rectangle is ceiled. That means that if the image space rectangle occupies less than one widget space pixel it is rounded up to one.

A use for this method would be if you for example is implementing an `IImageTool` and want to return a special `gtk.gdk.Cursor` if the pointer is over a hotspot in the image:

```
def cursor_at_point(self, x, y):
    wid_rect = self.view.image_to_widget_rect(self.hotspot)
    # Use the default cursor if the the view isn't realized
    # or if the pointer isn't over the hotspot.
    if not wid_rect:
        return None
    if not (x >= wid_rect.x and x < wid_rect.x + wid_rect.width and
           y >= wid_rect.y and y < wid_rect.y + wid_rect.height):
        return None
    # Use the hotspot cursor if the pointer is over the hotspot.
    return self.hotspot_cursor
```

Parameters

`rect`: a `gtk.gdk.Rectangle` in image space coordinates.

Return Value

a `gtk.gdk.Rectangle` of the widget space coordinates or `None`

9 Class `gtkimageview.PixbufDrawCache`

Cache that ensures fast redraws by storing the last draw operation. For example, when resizing an `ImageView`, it will receive an expose event and must redraw the damaged region. Unless fitting is `True`, most of the pixels it should draw are identical to the ones drawn the previous time. Redrawing them is wasteful because scaling and especially bilinear scaling is very slow. Therefore, `PixbufDrawCache` objectifies the drawing process and adds a cache with the last draw from which pixels can be fetched.

This class is present purely to ensure optimal speed. An `IImageTool` that is asked to redraw a part of the image view widget could either do it manually with something like this:

```
def paint_image(self, draw_opts, drawable):
    zoom_rect = draw_opts.zoom_rect
    scaled = draw_opts.pixbuf.scale(0, 0,
                                    zoom_rect.width, zoom_rect.height,
                                    -zoom_rect.x, -zoom_rect.y,
                                    draw_opts.zoom,
                                    draw_opts.interp,
                                    zoom_rect.x, zoom_rect.y,
                                    16,
                                    draw_opts.check_color1,
                                    draw_opts.check_color2)
    drawable.draw_pixbuf(NULL, scaled,
                        0, 0,
                        draw_opts.widget_x, draw_opts.widget_y,
                        zoom_rect.width, zoom_rect.height,
                        gtk.gdk.RGB_DITHER_MAX,
                        draw_opts.widget_x, draw_opts.widget_y)
```

9.1 Methods

<p><code>__init__(self)</code></p> <hr/> <p>Create a new pixbuf draw cache.</p>

invalidate(*self*)

Force the draw cache to scale the pixbuf at the next draw.

`PixbufDrawCache` tries to minimize the number of scale operations needed by caching the last drawn pixbuf. It would be inefficient to check the individual pixels inside the pixbuf so it assumes that if the memory address of the pixbuf has not changed, then the cache is good to use.

However, when the image data is modified, this assumption breaks, which is why this method must be used to tell draw cache about it. **See Also:** `DRAW_METHOD_SCALE`, `draw()`, `ImageView.damage_pixels()`

draw(*self*, *draw_opts*, *drawable*)

Draw on the drawable using the specified draw options. **Parameters**

`draw_opts`: `PixbufDrawOpts` to use in this draw.

`drawable`: a `gdk.Drawable` to draw on.

get_method(*cls*, *last_opts*, *new_opts*)

Get the fastest method to draw the specified draw options. `last_opts` is assumed to be the last `PixbufDrawOpts` used and `new_opts` is the one to use this time.

This function returns one of the three constants `DRAW_METHOD_CONTAINS`, `DRAW_METHOD_SCROLL` or `DRAW_METHOD_SCALE`. **Parameters**

`last_opts`: the last draw options used

`new_opts`: the current draw options

Return Value

the best draw method to use to draw

10 Class `gtkimageview.PixbufDrawOpts`

Container class which holds options for how the pixbuf should be drawn. Options include such things like the source rectangle in the pixbuf to draw, where to draw it, which zoom to use and so on. **See Also:** `PixbufDrawCache.draw()`

10.1 Methods

```
__init__(self, zoom, zoom_rect, widget_x, widget_y, interp, pixbuf, check_color1,
         check_color2)
```

10.2 Instance Variables

Name	Description
<code>zoom</code>	Zoom factor.
<code>zoom_rect</code>	Rectangle in zoom space coordinates of the area to draw.
<code>widget_x</code>	X-coordinate of the draw location on the widget.
<code>widget_y</code>	Y-coordinate of the draw location on the widget.
<code>interp</code>	Which <code>gtk.gdk.InterpType</code> to use.
<code>pixbuf</code>	Pixbuf to draw from.
<code>check_color1</code>	The first color to use for drawing transparent parts.
<code>check_color2</code>	The second color to use for drawing transparent parts.

Index

- gtkimageview (*module*), 2–6
 - gtkimageview.AnimationView (*class*), 7–9
 - gtkimageview.AnimationView.get_anim (*method*), 8
 - gtkimageview.AnimationView.get_is_playing (*method*), 8
 - gtkimageview.AnimationView.set_anim (*method*), 8
 - gtkimageview.AnimationView.set_is_playing (*method*), 8
 - gtkimageview.AnimationView.step (*method*), 8
 - gtkimageview.IImageTool (*class*), 10–11
 - gtkimageview.IImageTool.button_press (*method*), 10
 - gtkimageview.IImageTool.button_release (*method*), 10
 - gtkimageview.IImageTool.cursor_at_point (*method*), 11
 - gtkimageview.IImageTool.motion_notify (*method*), 10
 - gtkimageview.IImageTool.paint_image (*method*), 11
 - gtkimageview.IImageTool.pixbuf_changed (*method*), 10
 - gtkimageview.ImageNav (*class*), 12–13
 - gtkimageview.ImageNav.__init__ (*method*), 12
 - gtkimageview.ImageNav.get_pixbuf (*method*), 12
 - gtkimageview.ImageNav.grab (*method*), 12
 - gtkimageview.ImageNav.release (*method*), 12
 - gtkimageview.ImageNav.show_and_grab (*method*), 12
 - gtkimageview.ImageScrollWin (*class*), 14
 - gtkimageview.ImageScrollWin.__init__ (*method*), 14
 - gtkimageview.ImageToolDragger (*class*), 15
 - gtkimageview.ImageToolDragger.__init__ (*method*), 15
 - gtkimageview.ImageToolSelector (*class*), 16–18
 - gtkimageview.ImageToolSelector.__init__ (*method*), 17
 - gtkimageview.ImageToolSelector.get_selection (*method*), 17
 - gtkimageview.ImageToolSelector.set_selection (*method*), 18
 - gtkimageview.ImageToolSelector.sig_selection_changed (*method*), 17
 - gtkimageview.ImageView (*class*), 19–32
 - gtkimageview.ImageView.__init__ (*method*), 24
 - gtkimageview.ImageView.damage_pixels (*method*), 31
 - gtkimageview.ImageView.get_black_bg (*method*), 29
 - gtkimageview.ImageView.get_check_colors (*method*), 25
 - gtkimageview.ImageView.get_draw_rect (*method*), 25
 - gtkimageview.ImageView.get_fitting (*method*), 28
 - gtkimageview.ImageView.get_interpolation (*method*), 29
 - gtkimageview.ImageView.get_pixbuf (*method*), 27
 - gtkimageview.ImageView.get_show_cursor (*method*), 30
 - gtkimageview.ImageView.get_show_frame (*method*), 30
 - gtkimageview.ImageView.get_tool (*method*), 29
 - gtkimageview.ImageView.get_viewport (*method*), 25
 - gtkimageview.ImageView.get_zoom (*method*), 28
 - gtkimageview.ImageView.image_to_widget_rect (*method*), 31
 - gtkimageview.ImageView.set_black_bg (*method*), 29
 - gtkimageview.ImageView.set_fitting (*method*),

- 28
- gtkimageview.ImageView.set_interpolation (method), 30
- gtkimageview.ImageView.set_offset (method), 4
- 26
- gtkimageview.ImageView.set_pixbuf (method), 4
- 27
- gtkimageview.ImageView.set_show_cursor (method), 30
- gtkimageview.ImageView.set_show_frame (method), 30
- gtkimageview.ImageView.set_tool (method), 29
- gtkimageview.ImageView.set_transp (method), 26
- gtkimageview.ImageView.set_zoom (method), 28
- gtkimageview.ImageView.sig_pixbuf_changed (method), 24
- gtkimageview.ImageView.sig_zoom_changed (method), 24
- gtkimageview.ImageView.zoom_in (method), 31
- gtkimageview.ImageView.zoom_out (method), 31
- gtkimageview.library_version (function), 4
- gtkimageview.PixbufDrawCache (class), 33–34
- gtkimageview.PixbufDrawCache.__init__ (method), 33
- gtkimageview.PixbufDrawCache.draw (method), 34
- gtkimageview.PixbufDrawCache.get_method (class method), 34
- gtkimageview.PixbufDrawCache.invalidate (method), 33
- gtkimageview.PixbufDrawOpts (class), 35
- gtkimageview.PixbufDrawOpts.__init__ (method), 35
- gtkimageview.zooms_clamp_zoom (function), 5
- gtkimageview.zooms_get_max_zoom (function), 5
- gtkimageview.zooms_get_min_zoom (function), 5
- gtkimageview.zooms_get_zoom_in (function), 4
- gtkimageview.zooms_get_zoom_out (function), 4