Image(ID)

Image Data

The Image data-block is a shallow wrapper around image or video file(s) (on disk, as packed data, or generated).

All actual data like the pixel buffer, size, resolution etc. is cached in an `imbuf.types.ImBuf` image buffer (or several buffers in some cases, like UDIM textures, multi-views, animations…).

Several properties and functions of the Image data-block are then actually using/modifying its image buffer, and not the Image data-block itself.

Warning

One key limitation is that image buffers are not shared between different Image data-blocks, and they are not duplicated when copying an image.

So until a modified image buffer is saved on disk, duplicating its Image data-block will not propagate the underlying buffer changes to the new Image.

This example script generates an Image data-block with a given size, change its first pixel, rescale it, and duplicates the image.

The duplicated image still has the same size and colors as the original image at its creation, all editing in the original image's buffer is 'lost' in its copy.

```
import bpy

image_src = bpy.data.images.new('src', 1024, 102)
print(image_src.size)
print(image_src.pixels[0:4])

image_src.scale(1024, 720)
image_src.pixels[0:4] = (0.5, 0.5, 0.5, 0.5)
image_src.update()
print(image_src.size)
print(image_src.pixels[0:4])

image_dest = image_src.copy()
image_dest.update()
print(image_dest.size)
print(image_dest.pixels[0:4])
```

base classes — `bpy_struct`, `ID`

class bpy.types.**Image(ID)**

Image data-block referencing an external or packed image

alpha_mode

Representation of alpha in the image file, to convert to and from when saving and loading the image

- `STRAIGHT` Straight – Store RGB and alpha channels separately with alpha acting as a mask, also known as unassociated alpha. Commonly used by image editing applications and file formats like PNG..
- `PREMUL` Premultiplied – Store RGB channels with alpha multiplied in, also known as associated alpha. The natural format for renders an used by file formats like OpenEXR..
- `CHANNEL_PACKED` Channel Packed – Different images are packed in the RGB and alpha channels, and they should not affect each other. Channel packing is commonly used by game engines to save memory..
- `NONE` None – Ignore alpha channel from the file and make image fully opaque.

TYPE:

enum in ['STRAIGHT', 'PREMUL', 'CHANNEL_PACKED', 'NONE'], default 'STRAIGHT'

bindcode

OpenGL bindcode

TYPE:

int in [0, inf], default 0, (readonly)

channels

Number of channels in pixels buffer

TYPE:

int in [0, inf], default 0, (readonly)

colorspace_settings

Input color space settings

TYPE:

`ColorManagedInputColorspaceSettings`, (readonly)

depth

Image bit depth

TYPE:

int in [0, inf], default 0, (readonly)

display_aspect

Display Aspect for this image, does not affect rendering

TYPE:

`mathutils.Vector` of 2 items in [0.1, inf], default (1.0, 1.0)

file_format

Format used for re-saving this file

TYPE:

enum in Image Type Items, default 'TARGA'

filepath

Image/Movie file name

TYPE:

string, default "", (never None)

filepath_raw

Image/Movie file name (without data refreshing)

TYPE:

string, default "", (never None)

frame_duration

Duration (in frames) of the image (1 when not a video/sequence)

TYPE:

int in [0, inf], default 0, (readonly)

generated_color

Fill color for the generated image

TYPE:

TYPE:

float array of 4 items in [0, inf], default (0.0, 0.0, 0.0, 0.0)

generated_height

Generated image height

TYPE:

int in [1, 65536], default 1024

generated_type

Generated image type

TYPE:

enum in Image Generated Type Items, default 'UV_GRID'

generated_width

Generated image width

TYPE:

int in [1, 65536], default 1024

has_data

True if the image data is loaded into memory

TYPE:

boolean, default False, (readonly)

is_dirty

Image has changed and is not saved

TYPE:

boolean, default False, (readonly)

is_float

True if this image is stored in floating-point buffer

TYPE:

boolean, default False, (readonly)

is_multiview

Image has more than one view

TYPE:

boolean, default False, (readonly)

is_stereo_3d

Image has left and right views

TYPE:

boolean, default False, (readonly)

packed_file

First packed file of the image

TYPE:

`PackedFile`, (readonly)

packed_files

Collection of packed images

TYPE:

bpy_prop_collection of ImagePackedFile, (readonly)

pixels

Image buffer pixels in floating-point values

TYPE:

float in [-inf, inf], default 0.0

render_slots

Render slots of the image

TYPE:

RenderSlots bpy_prop_collection of RenderSlot, (readonly)

resolution

X/Y pixels per meter, for the image buffer

TYPE:

mathutils.Vector of 2 items in [-inf, inf], default (0.0, 0.0)

seam_margin

Margin to take into account when fixing UV seams during painting. Higher number would improve seam-fixes for mipmaps, but decreases performance.

TYPE:

int in [-32768, 32767], default 8

size

Width and height of the image buffer in pixels, zero when image data can't be loaded

TYPE:

int array of 2 items in [-inf, inf], default (0, 0), (readonly)

source

Where the image comes from

- FILE Single Image – Single image file.
- SEQUENCE Image Sequence – Multiple image files, as a sequence.
- MOVIE Movie – Movie file.
- GENERATED Generated – Generated image.
- VIEWER Viewer – Compositing node viewer.
- TILED UDIM Tiles – Tiled UDIM image texture.

TYPE:

enum in ['FILE', 'SEQUENCE', 'MOVIE', 'GENERATED', 'VIEWER', 'TILED'], default 'FILE'

stereo_3d_format

Settings for stereo 3d

TYPE:

Stereo3dFormat, (readonly, never None)

tiles

Tiles of the image

TYPE:

UDIMTiles bpy_prop_collection of UDIMTile, (readonly)

type

How to generate the image

TYPE:

enum in ['IMAGE', 'MULTILAYER', 'UV_TEST', 'RENDER_RESULT', 'COMPOSITING'], default 'IMAGE', (readonly)

use_deinterlace

Deinterlace movie file on load

TYPE:

boolean, default False

use_generated_float

Generate floating-point buffer

TYPE:

boolean, default False

use_half_precision

Use 16 bits per channel to lower the memory usage during rendering

TYPE:

boolean, default True

use_multiview

Use Multiple Views (when available)

TYPE:

boolean, default False

use_view_as_render

Apply render part of display transformation when displaying this image on the screen

TYPE:

boolean, default False

views_format

Mode to load image views

TYPE:

enum in Views Format Items, default 'INDIVIDUAL'

save_render(filepath, *, scene=None, quality=0)

Save image to a specific path using a scenes render settings

PARAMETERS:

- **filepath** (*string, (never None)*) – Output path
- **scene** (Scene, (optional)) – Scene to take image parameters from
- **quality** (*int in [0, 100], (optional)*) – Quality, Quality for image formats that support lossy compression, uses default quality if not specified

save(*, filepath='', quality=0, save_copy=False)

Save image

PARAMETERS:

PARAMETERS:

- **filepath** (*string, (optional, never None)*) – Output path, uses image data-block filepath if not specified
- **quality** (*int in [0, 100], (optional)*) – Quality, Quality for image formats that support lossy compression, uses default quality if not specified
- **save_copy** (*boolean, (optional)*) – Save Copy, Save the image as a copy, without updating current image's filepath

pack(*, data='', data_len=0)

Pack an image as embedded data into the .blend file

PARAMETERS:

- **data** (*byte string, (optional, never None)*) – data, Raw data (bytes, exact content of the embedded file)
- **data_len** (*int in [0, inf], (optional)*) – data_len, length of given data (mandatory if data is provided)

unpack(*, method='USE_LOCAL')

Save an image packed in the .blend file to disk

PARAMETERS:

method (enum in Unpack Method Items, (optional)) – method, How to unpack

reload()

Reload the image from its source path

update()

Update the display image from the floating-point buffer

scale(width, height, *, frame=0, tile_index=0)

Scale the buffer of the image, in pixels

PARAMETERS:

- **width** (*int in [1, inf]*) – Width
- **height** (*int in [1, inf]*) – Height
- **frame** (*int in [0, inf], (optional)*) – Frame, Frame (for image sequences)
- **tile_index** (*int in [0, inf], (optional)*) – Tile, Tile index (for tiled images)

gl_touch(*, frame=0, layer_index=0, pass_index=0)

Delay the image from being cleaned from the cache due inactivity

PARAMETERS:

- **frame** (*int in [0, inf], (optional)*) – Frame, Frame of image sequence or movie
- **layer_index** (*int in [0, inf], (optional)*) – Layer, Index of layer that should be loaded
- **pass_index** (*int in [0, inf], (optional)*) – Pass, Index of pass that should be loaded

RETURNS:

Error, OpenGL error value

RETURN TYPE:

int in [-inf, inf]

gl_load(*, frame=0, layer_index=0, pass_index=0)

Load the image into an OpenGL texture. On success, image.bindcode will contain the OpenGL texture bindcode. Colors read from the textur will be in scene linear color space and have premultiplied or straight alpha matching the image alpha mode.

PARAMETERS:

- **frame** (*int in [0, inf], (optional)*) – Frame, Frame of image sequence or movie
- **layer_index** (*int in [0, inf], (optional)*) – Layer, Index of layer that should be loaded
- **pass_index** (*int in [0, inf], (optional)*) – Pass, Index of pass that should be loaded

RETURNS:

Error, OpenGL error value

RETURN TYPE:

int in [-inf, inf]

gl_free()

Free the image from OpenGL graphics memory

filepath_from_user(*, image_user=None)

Return the absolute path to the filepath of an image frame specified by the image user

PARAMETERS:

image_user (`ImageUser` , (optional)) – Image user of the image to get filepath for

RETURNS:

File Path, The resulting filepath from the image and its user

RETURN TYPE:

string, (never None)

buffers_free()

Free the image buffers from memory

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

`bpy.types.Struct` subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- `bpy_struct.id_data`
- `ID.name`
- `ID.name_full`
- `ID.id_type`
- `ID.session_uid`
- `ID.is_evaluated`
- `ID.original`
- `ID.users`
- `ID.use_fake_user`
- `ID.use_extra_user`
- `ID.is_missing`
- `ID.is_runtime_data`
- `ID.is_editable`
- `ID.tag`
- `ID.is_library_indirect`
- `ID.library`
- `ID.library_weak_reference`
- `ID.asset_data`
- `ID.override_library`
- `ID.preview`

- ID.is_embedded_data

Inherited Functions

- bpy_struct.as_pointer
- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values
- ID.rename
- ID.evaluated_get
- ID.copy
- ID.asset_mark
- ID.asset_clear
- ID.asset_generate_preview
- ID.override_create
- ID.override_hierarchy_create
- ID.user_clear
- ID.user_remap
- ID.make_local
- ID.user_of_id
- ID.animation_data_create
- ID.animation_data_clear
- ID.update_tag
- ID.preview_ensure
- ID.bl_rna_get_subclass
- ID.bl_rna_get_subclass_py

References

- bpy.context.edit_image
- BlendData.images
- BlendDataImages.load
- BlendDataImages.new
- BlendDataImages.remove
- CameraBackgroundImage.image
- CompositorNodeCryptomatteV2.image
- CompositorNodeImage.image
- GeometryNodeInputImage.image
- ImagePaint.canvas
- ImagePaint.clone_image
- ImagePaint.stencil_image
- ImageTexture.image
- Material.texture_paint_images
- MaterialGPencilStyle.fill_image
- MaterialGPencilStyle.stroke_image
- MovieTrackingPlaneTrack.image
- NodeSocketImage.default_value
- NodeTreeInterfaceSocketImage.default_value
- PaintModeSettings.canvas_image
- ShaderNodeTexEnvironment.image
- ShaderNodeTexImage.image
- SpaceImageEditor.image
- TextureNodeImage.image
- UILayout.template_image_layers