

---

# **moderngl\_window Documentation**

***Release 2.0.4***

**Einar Forseiv**

**Nov 24, 2019**



# PROGRAMMING GUIDE

<b>1</b>	<b>Installation</b>	<b>3</b>
1.1	Installing with pip . . . . .	3
1.2	Optional dependencies . . . . .	3
1.3	Installing from source . . . . .	4
1.4	Running examples . . . . .	4
1.5	Running tests . . . . .	4
<b>2</b>	<b>Basic usage (WindowConfig)</b>	<b>5</b>
2.1	Basic example . . . . .	5
2.2	Resource loading . . . . .	6
2.3	Generic events and window types . . . . .	6
2.4	Command line arguments . . . . .	6
2.5	Window events . . . . .	7
2.6	Keyboard input . . . . .	7
2.7	Mouse input . . . . .	7
<b>3</b>	<b>Window Guide</b>	<b>9</b>
3.1	Using built in window types . . . . .	9
<b>4</b>	<b>Old Guide</b>	<b>11</b>
4.1	Register the moderngl.Context . . . . .	11
4.2	Register resource directories . . . . .	11
<b>5</b>	<b>Event Guide</b>	<b>13</b>
<b>6</b>	<b>The Resource System</b>	<b>15</b>
6.1	Resource types . . . . .	15
6.2	Resource paths . . . . .	15
6.3	Resource descriptions . . . . .	16
6.4	Loading resources . . . . .	16
6.4.1	Textures . . . . .	17
6.4.2	Programs . . . . .	17
6.4.3	Scenes . . . . .	17
6.4.4	Data . . . . .	17
<b>7</b>	<b>moderngl_window</b>	<b>19</b>
<b>8</b>	<b>moderngl_window.conf.Settings</b>	<b>21</b>
8.1	Methods . . . . .	21
8.2	Attributes . . . . .	23

<b>9</b>	<b>moderngl_window.context</b>	<b>27</b>
9.1	base.window.WindowConfig . . . . .	27
9.1.1	Methods . . . . .	28
9.1.2	Attributes . . . . .	31
9.2	base.BaseWindow . . . . .	32
9.2.1	Methods . . . . .	32
9.2.2	Attributes . . . . .	34
9.3	glfw.Window . . . . .	38
9.3.1	Methods . . . . .	38
9.3.2	Attributes . . . . .	39
9.3.3	Window Specific Methods . . . . .	44
9.4	headless.Window . . . . .	45
9.4.1	Methods . . . . .	45
9.4.2	Attributes . . . . .	46
9.5	pyglet.Window . . . . .	51
9.5.1	Methods . . . . .	51
9.5.2	Window Specific Methods . . . . .	52
9.5.3	Attributes . . . . .	54
9.6	pyqt5.Window . . . . .	58
9.6.1	Methods . . . . .	58
9.6.2	Window Specific Methods . . . . .	59
9.6.3	Attributes . . . . .	60
9.7	pyside2.Window . . . . .	65
9.7.1	Methods . . . . .	65
9.7.2	Window Specific Methods . . . . .	66
9.7.3	Attributes . . . . .	67
9.8	sdl2.Window . . . . .	71
9.8.1	Methods . . . . .	71
9.8.2	Window Specific Methods . . . . .	73
9.8.3	Attributes . . . . .	73
<b>10</b>	<b>moderngl_window.geometry</b>	<b>79</b>
<b>11</b>	<b>moderngl_window.loaders</b>	<b>81</b>
11.1	base.BaseLoader . . . . .	81
11.1.1	Method . . . . .	81
11.1.2	Attributes . . . . .	82
11.2	texture.t2d.Loader . . . . .	82
11.2.1	Method . . . . .	82
11.2.2	Attributes . . . . .	83
11.3	program.single.Loader . . . . .	83
11.3.1	Method . . . . .	83
11.3.2	Attributes . . . . .	84
11.4	program.separate.Loader . . . . .	85
11.4.1	Method . . . . .	85
11.4.2	Loader Specific Methods . . . . .	86
11.4.3	Attributes . . . . .	86
11.5	texture.array.Loader . . . . .	86
11.5.1	Method . . . . .	86
11.5.2	Attributes . . . . .	87
11.6	scene.wavefront.Loader . . . . .	87
11.6.1	Method . . . . .	87
11.6.2	Attributes . . . . .	88
11.7	scene.glTF2.Loader . . . . .	88

11.7.1	Method	88
11.7.2	Loader Specific Methods	89
11.7.3	Attributes	89
11.7.4	Loader Specific Attributes	89
11.8	scene.stl.Loader	90
11.8.1	Method	90
11.8.2	Attributes	90
11.9	data.json.Loader	91
11.9.1	Method	91
11.9.2	Attributes	91
11.10	data.text.Loader	92
11.10.1	Method	92
11.10.2	Attributes	92
11.11	data.binary.Loader	93
11.11.1	Method	93
11.11.2	Attributes	93
<b>12</b>	<b>moderngl_window.meta</b>	<b>95</b>
12.1	base.ResourceDescription	95
12.1.1	Methods	95
12.1.2	Attributes	95
12.2	texture.TextureDescription	96
12.2.1	Methods	96
12.2.2	Attributes	97
12.2.3	Inherited Attributes	97
12.3	program.ProgramDescription	98
12.3.1	Methods	99
12.3.2	Attributes	99
12.3.3	Inherited Attributes	100
12.4	scene.SceneDescription	100
12.4.1	Methods	101
12.4.2	Attributes	101
12.4.3	Inherited Attributes	101
12.5	data.DataDescription	102
12.5.1	Methods	103
12.5.2	Attributes	103
<b>13</b>	<b>moderngl_window.finders</b>	<b>105</b>
13.1	base.BaseFilesystemFinder	105
13.1.1	Methods	105
13.1.2	Attributes	105
13.2	texture.FilesystemFinder	105
13.2.1	Methods	105
13.2.2	Attributes	106
13.3	program.FilesystemFinder	106
13.3.1	Methods	106
13.3.2	Attributes	106
13.4	scene.FilesystemFinder	106
13.4.1	Methods	106
13.4.2	Attributes	106
13.5	data.FilesystemFinder	107
13.5.1	Methods	107
13.5.2	Attributes	107

<b>14 moderngl_window.opengl</b>	<b>109</b>
14.1 opengl.projection.Projection3D	109
14.1.1 Methods	109
14.1.2 Attributes	109
14.2 opengl.vao.VAO	110
14.2.1 Methods	111
14.2.2 Attributes	112
<b>15 moderngl_window.resources</b>	<b>113</b>
15.1 base.BaseRegistry	113
15.1.1 Methods	113
15.1.2 Attributes	114
15.2 textures.Textures	114
15.2.1 Methods	114
15.2.2 Attributes	115
15.3 programs.Programs	115
15.3.1 Methods	115
15.3.2 Attributes	116
15.4 scenes.Scenes	116
15.4.1 Methods	116
15.4.2 Attributes	117
15.5 base.DataFiles	117
15.5.1 Methods	117
15.5.2 Attributes	117
<b>16 moderngl_window.timers</b>	<b>119</b>
16.1 base.BaseTimer	119
16.1.1 Methods	119
16.1.2 Attributes	119
16.2 clock.Timer	120
16.2.1 Methods	120
16.2.2 Attributes	120
<b>17 moderngl_window.scene</b>	<b>121</b>
17.1 Camera	121
17.1.1 Methods	121
17.1.2 Attributes	122
17.2 KeyboardCamera	122
17.2.1 Methods	122
17.2.2 Attributes	124
17.3 Scene	125
17.3.1 Methods	125
17.3.2 Attributes	125
17.4 Node	126
17.4.1 Methods	126
17.4.2 Attributes	127
17.5 Mesh	127
17.5.1 Methods	127
17.6 Material	128
17.6.1 Methods	128
17.6.2 Attributes	128
17.7 MaterialTexture	129
17.7.1 Methods	129
17.7.2 Attributes	129

17.8	MeshProgram	129
17.8.1	Methods	129
17.8.2	Attributes	130
<b>18</b>	<b>Indices and tables</b>	<b>131</b>
	<b>Python Module Index</b>	<b>133</b>
	<b>Index</b>	<b>135</b>





A cross platform helper library for ModernGL making window creation and resource loading simple.

---

**Note:** Please report documentation improvements/issues on github. Writing documentation is difficult and we can't do it without you. Pull requests with documentation improvements are also greatly appreciated.

---



## INSTALLATION

### 1.1 Installing with pip

moderngl-window is available on PyPI:

```
pip install moderngl-window
```

### 1.2 Optional dependencies

We try to have as few requirements as possible and instead offer optional dependencies. You can create your own window types and loaders and don't want to force installing unnecessary dependencies.

By default we install pyglet as this is the default window type as it small and pretty much work out of the box on all platforms.

Optional dependencies for loaders:

```
# Wavefront / obj loading
pip install moderngl-window[pywavefront]
# STL loading
pip install moderngl-window[trimesh]
```

Installing dependencies for window types:

```
pip install moderngl-window[PySide2]
pip install moderngl-window[pyqt5]
pip install moderngl-window[glfw]
pip install moderngl-window[PySDL2]
```

Installing optional dependencies this way should ensure a compatible version is installed.

For glfw and sdl2 windows you also need install the library itself. Thees are also available as packages on linux and homebrew on OS X. For windows the DLLs can simply be placed in the root of your project.

- GLFW : <https://www.glfw.org/>
- SDL2 : <https://www.libsdl.org/download-2.0.php>

## 1.3 Installing from source

```
# clone repo (optionally clone over https)
git clone git@github.com:moderngl/moderngl-window.git
cd moderngl-window

# Create your virtualenv and activate
# We assume the user knows how to work with virtualenvs

# Install moderngl-window in editable mode
pip install -e .

# Install optional dev dependencies covering all window and loader types
pip install -r requirements.txt
```

Installing the package in editable mode will make you able to run tests and examples. We highly recommend using virtualenvs.

## 1.4 Running examples

Assuming you installed from source you should be able to run the examples in the *examples* directory directly after installing the dev requirements in the root of the project:

```
pip install -r requirements.txt
```

## 1.5 Running tests

Install test requirements:

```
pip install -r tests/requirements.txt
```

Run tests with tox:

```
# Run for specific environment
tox -e py35
tox -e py36
tox -e py37

# pep8 run
tox -e pep8

# Run all environments
tox
```

## BASIC USAGE (WINDOWCONFIG)

---

**Note:** This section is only relevant when using *WindowConfig*. Go to the Custom Usage section if you provide your own window and context or want more control.

---

Using the *WindowConfig* interface is the simplest way to start with moderngl-window. This can work for projects smaller projects and implies that this library provides the window and moderngl context.

The API docs for this class alone should cover a lot of ground, but we'll go through the basics here.

### 2.1 Basic example

The *WindowConfig* is simply a class you extend to customize/implement initialization, window parameters, rendering code, keyboard input, mouse input and access simpler shortcut methods for loading resources.

```
import moderngl_window as mglw

class Test(mglw.WindowConfig):
    gl_version = (3, 3)
    window_size = (1920, 1080)

    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        # Do initialization here
        self.prog = self.ctx.program(...)
        self.vao = self.ctx.vertex_array(...)
        self.texture = self.ctx.texture(self.wnd.size, 4)

    def render(self, time, frametime):
        # This method is called every frame
        self.vao.render()

# Blocking call entering rendering/event loop
mglw.run_window_config(Test)
```

The *WindowConfig* instance will by default receive three external instances in `__init__` that can be accessed later with `self`.

- `self.ctx`: The `moderngl.Context` created by the configured window type
- `self.wnd`: The window instance
- `self.timer`: The `moderngl_window.timers.clock.Timer` instance to control the current time (Values passed into `render`)

## 2.2 Resource loading

The *WindowConfig* class has built in shortcuts to the resource loading system.

```
self.load_texture('background.png')
self.load_texture_array('tiles.png', layers=16)
self.load_program('myprogram.glsl')
self.load_text('textfile.txt')
self.load_json('config.json')
self.load_binary('data.bin')
self.load_scene('cube.obj')
self.load_scene('city.gltf')
```

All paths used in resource loading are relative to an absolute path provided in the *WindowConfig*.

```
from pathlib import Path

class Test(mglw.WindowConfig):
    resource_dir = (Path(__file__).parent / 'resources').resolve()
```

If you need more than one search path for your resources, the *moderngl\_window.resources* module have methods for this.

## 2.3 Generic events and window types

The *WindowConfig* interface depends on the built in window types or a self-provided window implementation of *BaseWindow*. These window implementations converts window, key and mouse events into a unified system so the user can switch between different window types without altering the code.

Window libraries are not perfect and may at times work sub-optimally on some platforms. They might also have different performance profiles. The ability switch between window types by just changing a config value can be an advantage.

You can change what window class is used by passing in the `--window` option. Optionally you can modify the *WINDOW* attribute directly.

## 2.4 Command line arguments

The *run\_window\_config()* method also reads arguments from `sys.argv` making the user able to override config values in the class.

Example:

```
python test.py --window glfw --fullscreen --vsync --samples 16 --cursor false --size_
↪800x600
```

See code for *moderngl\_window.parse\_args()* for more details.

## 2.5 Window events

```
def resize(self, width: int, height: int):
    print("Window was resized. buffer size is {} x {}".format(width, height))

def close(self):
    print("The window is closing")

def iconify(self, iconify: bool):
    print("Window was iconified:", iconify)
```

## 2.6 Keyboard input

Implement the `key_event` and `unicode_char_entered` method to handle key events.

```
def key_event(self, key, action, modifiers):
    # Key presses
    if action == self.wnd.keys.ACTION_PRESS:
        if key == self.wnd.keys.SPACE:
            print("SPACE key was pressed")

        # Using modifiers (shift and ctrl)

        if key == self.wnd.keys.Z and modifiers.shift:
            print("Shift + Z was pressed")

        if key == self.wnd.keys.Z and modifiers.ctrl:
            print("ctrl + Z was pressed")

    # Key releases
    elif action == self.wnd.keys.ACTION_RELEASE:
        if key == self.wnd.keys.SPACE:
            print("SPACE key was released")

def unicode_char_entered(self, char: str):
    print('character entered:', char)
```

## 2.7 Mouse input

Implement the `mouse_*` methods to handle mouse input.

```
def mouse_position_event(self, x, y, dx, dy):
    print("Mouse position:", x, y, dx, dy)

def mouse_drag_event(self, x, y, dx, dy):
    print("Mouse drag:", x, y, dx, dy)

def mouse_scroll_event(self, x_offset: float, y_offset: float):
    print("Mouse wheel:", x_offset, y_offset)

def mouse_press_event(self, x, y, button):
    print("Mouse button {} pressed at {}, {}".format(button, x, y))
```

(continues on next page)

(continued from previous page)

```
def mouse_release_event(self, x: int, y: int, button: int):  
    print("Mouse button {} released at {}, {}".format(button, x, y))
```



## WINDOW GUIDE

We support the following window types:

- `pyglet`
- `glfw`
- `sdl2`
- `pyside2`
- `pyqt5`
- `headless`

### 3.1 Using built in window types

The library provides shortcuts for window creation in the `moderngl_window` module that will also handle context activation.

The `moderngl_window.conf.Settings` instance has sane default parameters for a window. See the `WINDOW` attribute.

```
import moderngl_window
from moderngl_window.conf import settings

settings.WINDOW['class'] = 'moderngl_window.context.glfw.Window'
settings.WINDOW['gl_version'] = (4, 1)
# ... etc ...

# Creates the window instance and activates its context
window = moderngl_window.create_window_from_settings()
```

There are more sane ways to apply different configuration values through convenient methods in the `Settings` class.

Window classes can of course also be instantiated manually if preferred, but this can generated a bit of extra work.

```
import moderngl_window

window_str = 'moderngl_window.context.pyglet.Window'
window_cls = moderngl_window.get_window_cls(window_str)
window = window_cls(
    title="My Window",
    gl_version=(4, 1),
    size=(1920, 1080),
```

(continues on next page)

(continued from previous page)

```
    ...  
)  
moderngl_window.activate_context(ctx=window.ctx)
```

You could also simply import the class directory and instantiate it, but that defeats the purpose of trying to be independent of a specific window library.

The rendering loop for build in windows are simple:

```
while not window.is_closing:  
    window.clear()  
    # Render stuff here  
    window.swap_buffers()
```

The `swap_buffers` method is important as it also pulls new input events for the next frame.

When not using a *WindowConfig* instance there are a few simple steps to get started.

## 4.1 Register the moderngl.Context

When not using the built in window types you need to at least tell moderngl\_window what your moderngl.Context is.

```
import moderngl
import moderngl_window

# Somewhere in your application a standalone or normal context is created
ctx = moderngl.create_standalone_context(require=330)
ctx = moderngl.create_context(require=330)

# Make sure you activate this context
moderngl_window.activate_context(ctx=ctx)
```

If there are no context activated the library will raise an exception when doing operations that requires one such as texture and scene loading.

When using the built in window types the context activation is normally done for you on creation.

## 4.2 Register resource directories

The resource loading system are using relative paths. These paths are relative one or multiple directories we registered in the resource system.

The *moderngl\_window.resources* module has methods for this.

```
from pathlib import Path
from moderngl_window import resources

# We recommend using pathlib
resources.register_dir(Path('absolute/path/to/resource/dir').resolve())
# .. but strings also works
resources.register_dir('absolute/path/to/resource/dir')
```

These needs to be absolute paths or an exception is raised. You can register as many paths as you want. The resource system will simply look for the file in every registered directory in the order they were added until it finds a match.

This library also supports separate search directories for shader programs, textures, scenes and various data files.



**EVENT GUIDE**

Work in progress



## THE RESOURCE SYSTEM

### 6.1 Resource types

The resource system has four different resource types/categories it can load.

- **Programs** : Shader programs (vertex, geometry, fragment, tessellation, compute)
- **Textures** : Textures of all different variations
- **Scenes**: Wavefront, GLTF2 and STL scenes/objects
- **Data**: A generic “lane” for everything else

Each of these resource categories have separate search directories, one or multiple loader classes and a *ResourceDescription* class we use to describe the resource we are loading with all its parameters.

### 6.2 Resource paths

Resources are loaded using relative paths. These paths are relative to one or multiple search directories we register using the *resources* module.

For simple usage were we have one or multiple resource directories with mixed resource types (programs, textures etc.) we can use the simplified version, *register\_dir()*.

```
from pathlib import Path
from moderngl_window import resources

# pathlib.Path (recommended)
resources.register_dir(Path('absoulte/path/using/pathlib'))

# Strings and/or os.path
resources.register_dir('absolute/string/path')
```

A resource finder system will scan through the registered directories in the order they were added loading the first resource found.

For more advanced usage were resources of different types are separated we can register resource type specific search directories:

- *register\_program\_dir()*
- *register\_texture\_dir()*
- *register\_scene\_dir()*
- *register\_data\_dir()*

This can be handy when dealing with larger quantities of files. These search directories are stored in the *Settings* instance and can for example be temporarily altered if needed. This means you can separate local and global resources in more complex situations. It could even be used to support themes by promoting a theme directory overriding global/default resources or some default theme directory.

## 6.3 Resource descriptions

Resource descriptions are basically just classes acting as bags of attributes describing the resource we are requesting. We have four standard classes.

- *ProgramDescription*
- *TextureDescription*
- *SceneDescription*
- *DataDescription*

Example:

```
from moderngl_window.meta import TextureDescription

# We are aiming to load wood.png horizontally flipped
# with generated mipmaps and high anisotropic filtering.
TextureDescription(
    path='wood.png',
    flip=True,
    mipmap=True,
    anisotropy=16.0,
)
```

New resource description classes can be created by extending the base *ResourceDescription* class. This is not uncommon when for example making a new loader class.

## 6.4 Loading resources

Now that we know about the different resource categories, search paths and resource descriptions, we're ready to actually load something.

Loading resources can in some situation be a bit verbose, but you can simplify by wrapping them in your own functions if needed. The *WindowConfig* class is already doing this and can be used as a reference.

```
from moderngl_window.resources import (
    textures,
    programs,
    scenes,
    data,
)
from moderngl_window.meta import (
    TextureDescription,
    ProgramDescription,
    SceneDescription,
    DataDescription,
)
```



### 6.4.1 Textures

```
# Load a 2D texture
texture = textures.load(TextureDescription(path='wood.png'))

# Load wood.png horizontally flipped with generated mipmaps and high anisotropic
↳ filtering.
textures.load(TextureDescription(path='wood.png', flip=True, mipmap=True,
↳ anisotropy=16.0))

# Load a texture array containing 10 vertically stacked tile textures
textures.load(TextureDescription(path='tiles.png', layers=10, mipmap=True,
↳ anisotropy=8.0))
```

### 6.4.2 Programs

```
# Load a shader program in a single glsl file
program = programs.load(ProgramDescription(path='fun.glsl'))

# Load a shader program from multiple glsl files
program = programs.load(
    ProgramDescription(
        vertex_shader='sphere_vert.glsl',
        geometry_shader='sphere_geo.glsl',
        fragment_shader='sphere_fs.glsl',
    )
)
```

### 6.4.3 Scenes

```
# Load a GLTF2 scene
scene = scenes.load(SceneDescription(path="city.gltf"))

# Load a wavefront scene
scene = scenes.load(SceneDescription(path="earth.obj"))

# Load an STL file
scene = scenes.load(SceneDescription(path="apollo_landing_site_18.stl"))
```

### 6.4.4 Data

```
# Load text file
text = data.load(DataDescription(path='notes.txt'))

# Load config file as a dict
config_dict = data.load(DataDescription(path='config.json'))

# Load binary data
data = data.load(DataDescription(path='data.bin', kind='binary'))
```

For more information about supported parameters see the [api documentation](#).



## MODERNGL\_WINDOW

General helper functions aiding in the bootstrapping of this library.

`moderngl_window.setup_basic_logging(level: int)`

Set up basic logging

**Parameters** `level` (*int*) – The log level

`moderngl_window.activate_context(window: moderngl_window.context.base.window.BaseWindow  
= None, ctx: moderngl.context.Context = None)`

Register the active window and context. If only a window is supplied the context is taken from the window. Only a context can also be passed in.

### Keyword Arguments

- **window** (*window*) – The window to activate
- **ctx** (*moderngl.Context*) – The moderngl context to activate

`moderngl_window.window()`

Obtain the active window

`moderngl_window.ctx()`

Obtain the active context

`moderngl_window.get_window_cls(window: str = None) →  
Type[moderngl_window.context.base.window.BaseWindow]`

Attempt to obtain a window class using the full dotted python path. This can be used to import custom or modified window classes.

**Parameters** `window` (*str*) – Name of the window

**Returns** A reference to the requested window class. Raises exception if not found.

`moderngl_window.get_local_window_cls(window: str = None) →  
Type[moderngl_window.context.base.window.BaseWindow]`

Attempt to obtain a window class in the moderngl\_window package using short window names such as `pyglet` or `glfw`.

**Parameters** `window` (*str*) – Name of the window

**Returns** A reference to the requested window class. Raises exception if not found.

`moderngl_window.find_window_classes() → List[str]`

Find available window packages

**Returns** A list of available window packages

`moderngl_window.create_window_from_settings()` → `moderngl_window.context.base.window.BaseWindow`  
Creates a window using configured values in `moderngl_window.conf.Settings.WINDOW`. This will also activate the window/context.

**Returns** The Window instance

`moderngl_window.run_window_config` (*config\_cls: moderngl\_window.context.base.window.WindowConfig*,  
*timer=None, args=None*) → None  
Run an WindowConfig entering a blocking main loop

**Parameters**

- **config\_cls** – The WindowConfig class to render
- **args** – Override sys.args

`moderngl_window.parse_args` (*args=None*)  
Parse arguments from sys.argv

## MODERNGL\_WINDOW.CONF.SETTINGS

### `moderngl_window.conf.Settings`

Bag of settings values. New attributes can be freely added runtime. Various `apply*` methods are supplied so the user have full control over how settings values are initialized. This is especially useful for more custom usage. And instance of the *Settings* class is created when the *conf* module is imported.

Attribute names must currently be in upper case to be recognized.

Some examples of usage:

```
from moderngl_window.conf import settings

# Mandatory settings values
try:
    value = settings.VALUE
except KeyError:
    raise ValueError("This settings value is required")

# Fallback in code
value = getattr(settings, 'VALUE', 'default_value')

# Pretty printed string representation for easy inspection
print(settings)
```

## 8.1 Methods

### `Settings.__init__()`

Initialize settings with default values

### `Settings.apply_default_settings()` → None

Apply keys and values from the default settings module located in this package. This is to ensure we always have the minimal settings for the system to run.

If replacing or customizing the settings class you must always apply default settings to ensure compatibility when new settings are added.

### `Settings.apply_settings_from_env()` → None

Apply settings from `MODERNGL_WINDOW_SETTINGS_MODULE` environment variable. If the environment variable is undefined no action will be taken. Normally this would be used to easily be able to switch between different configuration by setting env vars before executing the program.

Example:

```
import os
from moderngl_window.conf import settings

os.environ['MODERNGL_WINDOW_SETTINGS_MODULE'] = 'python.path.to.module'
settings.apply_settings_from_env()
```

**Raises ImproperlyConfigured if the module was not found –**

`Settings.apply_from_module_name(settings_module_name: str) → None`

Apply settings from a python module by supplying the full pythonpath to the module.

**Parameters** `settings_module_name` (*str*) – Full python path to the module

**Raises ImproperlyConfigured if the module was not found –**

`Settings.apply_from_dict(data: dict) → None`

Apply settings values from a dictionary

Example:

```
>> from moderngl_window.conf import settings
>> settings.apply_dict({'SOME_VALUE': 1})
>> settings.SOME_VALUE
1
```

`Settings.apply_from_module(module: module) → None`

Apply settings values from a python module

Example:

```
my_settings.py module containing the following line:
SOME_VALUE = 1

>> from moderngl_window.conf import settings
>> import my_settings
>> settings.apply_module(my_settings)
>> settings.SOME_VALUE
1
```

`Settings.apply_from_cls(cls) → None`

Apply settings values from a class namespace

Example:

```
>> from moderngl_window.conf import settings
>> class MySettings:
>>     SOME_VALUE = 1
>>
>> settings.apply(MySettings)
>> settings.SOME_VALUE
1
```

`Settings.apply_from_iterable(iterable: Union[collections.abc.Iterable, generator]) → None`

Apply (key, value) pairs from an iterable or generator

`Settings.to_dict()`

Create a dict representation of the settings Only uppercase attributes are included

**Returns** dict representation

**Return type** dict

## 8.2 Attributes

### Settings.WINDOW

Window/screen properties. Most importantly the `class` attribute decides what class should be used to handle the window.

```
# Default values
WINDOW = {
    "gl_version": (3, 3),
    "class": "moderngl_window.context.pyglet.Window",
    "size": (1280, 720),
    "aspect_ratio": 16 / 9,
    "fullscreen": False,
    "resizable": True,
    "title": "ModernGL Window",
    "vsync": True,
    "cursor": True,
    "samples": 0,
}
```

Other Properties:

- `gl_version`: The minimum required major/minor OpenGL version
- `size`: The window size to open.
- `aspect_ratio` is the enforced aspect ratio of the viewport.
- `fullscreen`: True if you want to create a context in fullscreen mode
- `resizable`: If the window should be resizable. This only applies in windowed mode.
- `vsync`: Only render one frame per screen refresh
- `title`: The visible title on the window in windowed mode
- `cursor`: Should the mouse cursor be visible on the screen? Disabling this is also useful in windowed mode when controlling the camera on some platforms as moving the mouse outside the window can cause issues.
- `Samples`: Number if samples used in multisampling. Values above 1 enables multisampling.

The created window frame buffer will by default use:

- RGBA8 (32 bit per pixel)
- 24 bit depth buffer
- Double buffering
- color and depth buffer is cleared for every frame

### Settings.SCREENSHOT\_PATH

Absolute path to the directory screenshots will be saved by the screenshot module. Screenshots will end up in the project root of not defined. If a path is configured, the directory will be auto-created.

### Settings.PROGRAM\_FINDERS

Finder classes for locating programs/shaders.

```
# Default values
PROGRAM_FINDERS = [
    "moderngl_window.finders.program.FileSystemFinder",
]
```

**Settings.TEXTURE\_FINDERS**

Finder classes for locating textures.

```
# Default values
TEXTURE_FINDERS = [
    "moderngl_window.finders.texture.FileSystemFinder",
]
```

**Settings.SCENE\_FINDERS**

Finder classes for locating scenes.

```
# Default values
SCENE_FINDERS = [
    "moderngl_window.finders.scene.FileSystemFinder",
]
```

**Settings.DATA\_FINDERS**

Finder classes for locating data files.

```
# Default values
DATA_FINDERS = [
    "moderngl_window.finders.data.FileSystemFinder",
]
```

**Settings.PROGRAM\_DIRS**

Lists of *str* or *pathlib.Path* used by `FileSystemFinder` to look for programs/shaders.

**Settings.TEXTURE\_DIRS**

Lists of *str* or *pathlib.Path* used by `FileSystemFinder` to look for textures.

**Settings.SCENE\_DIRS**

Lists of *str* or *pathlib.Path* used by `FileSystemFinder` to look for scenes (obj, gltf, stl etc).

**Settings.DATA\_DIRS**

Lists of *str* or *pathlib.Path* used by `FileSystemFinder` to look for data files.

**Settings.PROGRAM\_LOADERS**

Classes responsible for loading programs/shaders.

```
# Default values
PROGRAM_LOADERS = [
    'moderngl_window.loaders.program.single.Loader',
    'moderngl_window.loaders.program.separate.Loader',
]
```

**Settings.TEXTURE\_LOADERS**

Classes responsible for loading textures.

```
# Default values
TEXTURE_LOADERS = [
    'moderngl_window.loaders.texture.t2d.Loader',
    'moderngl_window.loaders.texture.array.Loader',
]
```



**Settings.** **SCENE\_LOADERS**

Classes responsible for loading scenes.

```
# Default values
SCENE_LOADERS = [
    "moderngl_window.loaders.scene.gltf.GLTF2",
    "moderngl_window.loaders.scene.wavefront.ObjLoader",
    "moderngl_window.loaders.scene.stl_loader.STLLoader",
]
```

**Settings.** **DATA\_LOADERS**

Classes responsible for loading data files.

```
# Default values
DATA_LOADERS = [
    'moderngl_window.loaders.data.binary.Loader',
    'moderngl_window.loaders.data.text.Loader',
    'moderngl_window.loaders.data.json.Loader',
]
```



## MODERNGL\_WINDOW.CONTEXT

### 9.1 base.window.WindowConfig

`moderngl_window.context.base.window.WindowConfig`

Creating a `WindowConfig` instance is the simplest interface this library provides to open and window, handle inputs and provide simple shortcut method for loading basic resources. It's appropriate for projects with basic needs.

Example:

```
import moderngl_window

class MyConfig(moderngl_window.WindowConfig):
    gl_version = (3, 3)
    window_size = (1920, 1080)
    aspect_ratio = 16 / 9
    title = "My Config"
    resizable = False
    samples = 8

    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        # Do other initialization here

    def render(self, time: float, frametime: float):
        # Render stuff here with ModernGL

    def resize(self, width: int, height: int):
        print("Window was resized. buffer size is {} x {}".format(width, height))

    def mouse_position_event(self, x, y, dx, dy):
        print("Mouse position:", x, y)

    def mouse_press_event(self, x, y, button):
        print("Mouse button {} pressed at {}, {}".format(button, x, y))

    def mouse_release_event(self, x: int, y: int, button: int):
        print("Mouse button {} released at {}, {}".format(button, x, y))

    def key_event(self, key, action, modifiers):
        print(key, action, modifiers)
```

### 9.1.1 Methods

WindowConfig.\_\_init\_\_(ctx: moderngl.context.Context = None, wnd: moderngl\_window.context.base.window.BaseWindow = None, timer: moderngl\_window.timers.base.BaseTimer = None, \*\*kwargs)

Initialize the window config

#### Keyword Arguments

- **ctx** (*moderngl.Context*) – The moderngl context
- **wnd** – The window instance
- **timer** – The timer instance

WindowConfig.render(time: float, frame\_time: float)

Renders the assigned effect

#### Parameters

- **time** (*float*) – Current time in seconds
- **frame\_time** (*float*) – Delta time from last frame in seconds

WindowConfig.resize(width: int, height: int)

Called every time the window is resized in case the we need to do internal adjustments.

#### Parameters

- **width** (*int*) – width in buffer size (not window size)
- **height** (*int*) – height in buffer size (not window size)

WindowConfig.close()

Called when the window is closed

WindowConfig.key\_event(key: Any, action: Any, modifiers: moderngl\_window.context.base.keys.KeyModifiers)

Called for every key press and release. Depending on the library used, key events may trigger repeating events during the pressed duration based on the configured key repeat on the users operating system.

#### Parameters

- **key** – The key that was press. Compare with self.wnd.keys.
- **action** – self.wnd.keys.ACTION\_PRESS or ACTION\_RELEASE
- **modifiers** – Modifier state for shift and ctrl

WindowConfig.mouse\_position\_event(x: int, y: int, dx: int, dy: int)

Reports the current mouse cursor position in the window

#### Parameters

- **x** (*int*) – X postion of the mouse cursor
- **y** (*int*) – Y position of the mouse cursor
- **dx** (*int*) – X delta postion
- **dy** (*int*) – Y delta position

WindowConfig.mouse\_press\_event(x: int, y: int, button: int)

Called when a mouse button in pressed

#### Parameters

- **x** (*int*) – X position the press occurred

- **y** (*int*) – Y position the press occurred
- **button** (*int*) – 1 = Left button, 2 = right button

WindowConfig.**mouse\_release\_event** (*x: int, y: int, button: int*)

Called when a mouse button is released

#### Parameters

- **x** (*int*) – X position the release occurred
- **y** (*int*) – Y position the release occurred
- **button** (*int*) – 1 = Left button, 2 = right button

WindowConfig.**mouse\_drag\_event** (*x: int, y: int, dx: int, dy: int*)

Called when the mouse is moved while a button is pressed.

#### Parameters

- **x** (*int*) – X position of the mouse cursor
- **y** (*int*) – Y position of the mouse cursor
- **dx** (*int*) – X delta position
- **dy** (*int*) – Y delta position

WindowConfig.**mouse\_scroll\_event** (*x\_offset: float, y\_offset: float*)

Called when the mouse wheel is scrolled.

Some input devices also support horizontal scrolling, but vertical scrolling is fairly universal.

#### Parameters

- **x\_offset** (*int*) – X scroll offset
- **y\_offset** (*int*) – Y scroll offset

WindowConfig.**unicode\_char\_entered** (*char: str*)

Called when the user entered a unicode character.

**Parameters** **char** (*str*) – The character entered

WindowConfig.**load\_texture\_2d** (*path: str, flip=True, mipmap=False, mipmap\_levels: Tuple[int, int] = None, anisotropy=1.0, \*\*kwargs*) → moderngl.texture.Texture

Loads a 2D texture

**Parameters** **path** (*str*) – Path to the texture relative to search directories

#### Keyword Arguments

- **flip** (*boolean*) – Flip the image horizontally
- **mipmap** (*bool*) – Generate mipmaps. Will generate max possible levels unless *mipmap\_levels* is defined.
- **mipmap\_levels** (*tuple*) – (base, max\_level) controlling mipmap generation. When defined the *mipmap* parameter is automatically *True*
- **anisotropy** (*float*) – Number of samples for anisotropic filtering
- **\*\*kwargs** – Additional parameters to TextureDescription

**Returns** Texture instance

**Return type** moderngl.Texture

WindowConfig.**load\_texture\_array**(*path*: str, *layers*: int = 0, *flip*=True, *mipmap*=False, *mipmap\_levels*: Tuple[int, int] = None, *anisotropy*=1.0, *\*\*kwargs*) → moderngl.texture\_array.TextureArray

Loads a texture array.

**Parameters** *path* (str) – Path to the texture relative to search directories

**Keyword Arguments**

- **layers** (int) – How many layers to split the texture into vertically
- **flip** (boolean) – Flip the image horizontally
- **mipmap** (bool) – Generate mipmaps. Will generate max possible levels unless *mipmap\_levels* is defined.
- **mipmap\_levels** (tuple) – (base, max\_level) controlling mipmap generation. When defined the *mipmap* parameter is automatically *True*
- **anisotropy** (float) – Number of samples for anisotropic filtering
- **\*\*kwargs** – Additional parameters to TextureDescription

**Returns** The texture instance

**Return type** moderngl.TextureArray

WindowConfig.**load\_program**(*path*=None, *vertex\_shader*=None, *geometry\_shader*=None, *fragment\_shader*=None, *tess\_control\_shader*=None, *tess\_evaluation\_shader*=None) → moderngl.program.Program

Loads a shader program.

Note that *path* should only be used if all shaders are defined in the same glsl file separated by defines.

**Keyword Arguments**

- **path** (str) – Path to a single glsl file
- **vertex\_shader** (str) – Path to vertex shader
- **geometry\_shader** (str) – Path to geometry shader
- **fragment\_shader** (str) – Path to fragment shader
- **tess\_control\_shader** (str) – Path to tessellation control shader
- **tess\_evaluation\_shader** (str) – Path to tessellation eval shader

**Returns** The program instance

**Return type** moderngl.Program

WindowConfig.**load\_text**(*path*: str, *\*\*kwargs*) → str

Load a text file.

**Parameters**

- **path** (str) – Path to the file relative to search directories
- **\*\*kwargs** – Additional parameters to DataDescription

**Returns** Contents of the text file

**Return type** str

WindowConfig.**load\_json**(*path*: str, *\*\*kwargs*) → dict

Load a json file

**Parameters**

- **path** (*str*) – Path to the file relative to search directories
- **\*\*kwargs** – Additional parameters to DataDescription

**Returns** Contents of the json file

**Return type** dict

WindowConfig.**load\_binary** (*path: str, \*\*kwargs*) → bytes  
Load a file in binary mode.

**Parameters**

- **path** (*str*) – Path to the file relative to search directories
- **\*\*kwargs** – Additional parameters to DataDescription

**Returns** The byte data of the file

**Return type** bytes

WindowConfig.**load\_scene** (*path: str, cache=False, attr\_names=<class 'moderngl\_window.geometry.attributes.AttributeNames'>, kind=None, \*\*kwargs*) → moderngl\_window.scene.scene.Scene

Loads a scene.

**Keyword Arguments**

- **path** (*str*) – Path to the file relative to search directories
- **cache** (*str*) – Use the loader caching system if present
- **attr\_names** (*AttributeNames*) – Attrib name config
- **kind** (*str*) – Override loader kind
- **\*\*kwargs** – Additional parameters to SceneDescription

**Returns** The scene instance

**Return type** Scene

## 9.1.2 Attributes

WindowConfig.**window\_size**  
Size of the window.

```
# Default value
window_size = (1280, 720)
```

WindowConfig.**resizable**  
Determines if the window should be resizable

```
# Default value
resizable = True
```

WindowConfig.**gl\_version**  
The minimum required OpenGL version required

```
# Default value
gl_version = (3, 3)
```

WindowConfig.**title**

Title of the window

```
# Default value
title = "Example"
```

WindowConfig.**aspect\_ratio**

The enforced aspect ratio of the viewport. When specified back borders will be calculated both vertically and horizontally if needed.

This property can be set to `None` to disable the fixed viewport system.

```
# Default value
aspect_ratio = 16 / 9
```

WindowConfig.**cursor**

Determines if the mouse cursor should be visible inside the window. If enabled on some platforms

```
# Default value
cursor = True
```

WindowConfig.**samples**

Number of samples to use in multisampling.

```
# Default value
samples = 4
```

WindowConfig.**resource\_dir**

Absolute path to your resource directory containing textures, scenes, shaders/programs or data files. The `load_` methods in this class will look for resources in this path. This attribute can be a `str` or a `pathlib.Path`.

```
# Default value
resource_dir = None
```

WindowConfig.**log\_level**

Sets the log level for this library using the standard *logging* module.

```
# Default value
log_level = logging.INFO
```

## 9.2 base.BaseWindow

### 9.2.1 Methods

BaseWindow.**\_\_init\_\_**(*title='ModernGL', gl\_version=(3, 3), size=(1280, 720), resizable=True, fullscreen=False, vsync=True, aspect\_ratio: float = None, samples=4, cursor=True, \*\*kwargs*)

Initialize a window instance.

#### Parameters

- **title** (*str*) – The window title
- **gl\_version** (*tuple*) – Major and minor version of the opengl context to create
- **size** (*tuple*) – Window size x, y



- **resizable** (*bool*) – Should the window be resizable?
- **fullscreen** (*bool*) – Open window in fullscreen mode
- **vsync** (*bool*) – Enable/disable vsync
- **aspect\_ratio** (*float*) – The desired fixed aspect ratio. Can be set to `None` to make aspect ratio be based on the actual window size.
- **samples** (*int*) – Number of MSAA samples for the default framebuffer
- **cursor** (*bool*) – Enable/disable displaying the cursor inside the window

`BaseWindow.init_mgl_context()` → `None`

Create or assign a ModernGL context. If no context is supplied a context will be created using the window's `gl_version`.

**Keyword Arguments** `ctx` – An optional custom ModernGL context

`BaseWindow.is_key_pressed(key)` → `bool`

Returns: The press state of a key

`BaseWindow.close()` → `None`

Signal for the window to close

`BaseWindow.use()`

Bind the window's framebuffer

`BaseWindow.clear(red=0.0, green=0.0, blue=0.0, alpha=0.0, depth=1.0, viewport=None)`

Binds and clears the default framebuffer

#### Parameters

- **red** (*float*) – color component
- **green** (*float*) – color component
- **blue** (*float*) – color component
- **alpha** (*float*) – alpha component
- **depth** (*float*) – depth value
- **viewport** (*tuple*) – The viewport

`BaseWindow.render(time=0.0, frame_time=0.0)` → `None`

Renders a frame by calling the configured render callback

#### Keyword Arguments

- **time** (*float*) – Current time in seconds
- **frame\_time** (*float*) – Delta time from last frame in seconds

`BaseWindow.swap_buffers()` → `None`

Library specific buffer swap method. Must be overridden.

`BaseWindow.resize(width, height)` → `None`

Should be called every time window is resized so the example can adapt to the new size if needed

`BaseWindow.destroy()` → `None`

A library specific destroy method is required

`BaseWindow.set_default_viewport()` → `None`

Calculates the and sets the viewport based on window configuration.

The viewport will based on the configured fixed aspect ratio if set. If no fixed aspect ratio is set the viewport will be scaled to the entire window size regardless of size.

Will add black borders and center the viewport if the window do not match the configured viewport (fixed only)

`BaseWindow.print_context_info()`  
Prints moderngl context info.

## 9.2.2 Attributes

`BaseWindow.keys`  
Window specific key constants

`BaseWindow.ctx`  
The ModernGL context for the window  
**Type** `moderngl.Context`

`BaseWindow.fbo`  
The default framebuffer  
**Type** `moderngl.Framebuffer`

`BaseWindow.title`  
Window title.  
This property can also be set:

```
window.title = "New Title"
```

**Type** `str`

`BaseWindow.gl_version`  
(major, minor) required OpenGL version  
**Type** `Tuple[int, int]`

`BaseWindow.width`  
The current window width  
**Type** `int`

`BaseWindow.height`  
The current window height  
**Type** `int`

`BaseWindow.size`  
current window size.

This property also support assignment:

```
# Resize the window to 1000 x 1000
window.size = 1000, 1000
```

**Type** `Tuple[int, int]`

`BaseWindow.position`  
The current window position.

This property can also be set to move the window:

```
# Move window to 100, 100
window.position = 100, 100
```

**Type** Tuple[int, int]

BaseWindow.**buffer\_width**  
the current window buffer width

**Type** int

BaseWindow.**buffer\_height**  
the current window buffer height

**Type** int

BaseWindow.**buffer\_size**  
tuple with the current window buffer size

**Type** Tuple[int, int]

BaseWindow.**pixel\_ratio**  
The framebuffer/window size ratio

**Type** float

BaseWindow.**viewport**  
current window viewport

**Type** Tuple[int, int, int, int]

BaseWindow.**viewport\_size**  
Size of the viewport.  
Equivalent to `self.viewport[2], self.viewport[3]`

**Type** Tuple[int,int]

BaseWindow.**viewport\_width**  
The width of the viewport.  
Equivalent to `self.viewport[2].`

**Type** int

BaseWindow.**viewport\_height**  
The height of the viewport  
Equivalent to `self.viewport[3].`

**Type** int

BaseWindow.**frames**  
Number of frames rendered

**Type** int

BaseWindow.**resizable**  
Window is resizable

**Type** bool

BaseWindow.**fullscreen**  
Window is in fullscreen mode

**Type** bool

#### BaseWindow.config

Get the current WindowConfig instance

This property can also be set. Assigning a WindowConfig instance will automatically set up the necessary event callback methods:

```
window.config = window_config_instance
```

#### BaseWindow.vsync

vertical sync enabled/disabled

**Type** bool

#### BaseWindow.aspect\_ratio

The current aspect ratio of the window. If a fixed aspect ratio was passed to the window initializer this value will always be returned. Otherwise width / height will be returned.

This property is read only.

**Type** float

#### BaseWindow.fixed\_aspect\_ratio

The fixed aspect ratio for the window.

Can be set to None to disable fixed aspect ratio making the aspect ratio adjust to the actual window size

This will affects how the viewport is calculated and the reported value from the aspect\_ratio property:

```
# Enabled fixed aspect ratio
window.fixed_aspect_ratio = 16 / 9

# Disable fixed aspect ratio
window.fixed_aspect_ratio = None
```

**Type** float

#### BaseWindow.samples

Number of Multisample anti-aliasing (MSAA) samples

**Type** float

#### BaseWindow.cursor

Should the mouse cursor be visible inside the window?

This property can also be assigned to:

```
# Disable cursor
window.cursor = False
```

**Type** bool

#### BaseWindow.mouse\_exclusivity

If mouse exclusivity is enabled.

When you enable mouse-exclusive mode, the mouse cursor is no longer available. It is not merely hidden – no amount of mouse movement will make it leave your application. This is for example useful when you don't want the mouse leaving the screen when rotating a 3d scene.

This property can also be set:

```
window.mouse_exclusivity = True
```

**Type** bool

`BaseWindow.render_func`

The render callable

This property can also be used to assign a callable.

**Type** callable

`BaseWindow.resize_func`

Get or set the resize callable

**Type** callable

`BaseWindow.close_func`

Get or set the close callable

**Type** callable

`BaseWindow.iconify_func`

Get or set the iconify/show/hide callable

**Type** callable

`BaseWindow.key_event_func`

Get or set the key\_event callable

**Type** callable

`BaseWindow.mouse_position_event_func`

Get or set the mouse\_position callable

**Type** callable

`BaseWindow.mouse_press_event_func`

Get or set the mouse\_press callable

**Type** callable

`BaseWindow.mouse_release_event_func`

Get or set the mouse\_release callable

**Type** callable

`BaseWindow.mouse_drag_event_func`

Get or set the mouse\_drag callable

**Type** callable

`BaseWindow.mouse_scroll_event_func`

Get or set the mouse\_scroll\_event callable

**Type** callable

`BaseWindow.unicode_char_entered_func`

Get or set the unicode\_char\_entered callable

**Type** callable

`BaseWindow.is_closing`

Is the window about to close?

**Type** bool

`BaseWindow.mouse = <class 'moderngl_window.context.base.window.MouseButtons'>`  
 Mouse button enum

`BaseWindow.mouse_states`  
 Mouse button state structure.

The current mouse button states.

```

window.mouse_buttons.left
window.mouse_buttons.right
window.mouse_buttons.middle
    
```

**Type** MouseButtonStates

`BaseWindow.modifiers`  
 (KeyModifiers) The current keyboard modifiers

`BaseWindow.gl_version_code`  
 Generates the version code integer for the selected OpenGL version.

`gl_version (4, 1)` returns 410

**Type** int

## 9.3 glfw.Window

### 9.3.1 Methods

`Window.__init__ (**kwargs)`  
 Initialize a window instance.

#### Parameters

- **title** (*str*) – The window title
- **gl\_version** (*tuple*) – Major and minor version of the opengl context to create
- **size** (*tuple*) – Window size x, y
- **resizable** (*bool*) – Should the window be resizable?
- **fullscreen** (*bool*) – Open window in fullscreen mode
- **vsync** (*bool*) – Enable/disable vsync
- **aspect\_ratio** (*float*) – The desired fixed aspect ratio. Can be set to `None` to make aspect ratio be based on the actual window size.
- **samples** (*int*) – Number of MSAA samples for the default framebuffer
- **cursor** (*bool*) – Enable/disable displaying the cursor inside the window

`Window.init_mgl_context ()` → None  
 Create or assign a ModernGL context. If no context is supplied a context will be created using the window's `gl_version`.

**Keyword Arguments** `ctx` – An optional custom ModernGL context

`Window.is_key_pressed (key)` → bool  
 Returns: The press state of a key

`Window.close()` → None

Suggest to glfw the window should be closed soon

`Window.use()`

Bind the window's framebuffer

`Window.clear(red=0.0, green=0.0, blue=0.0, alpha=0.0, depth=1.0, viewport=None)`

Binds and clears the default framebuffer

#### Parameters

- **red** (*float*) – color component
- **green** (*float*) – color component
- **blue** (*float*) – color component
- **alpha** (*float*) – alpha component
- **depth** (*float*) – depth value
- **viewport** (*tuple*) – The viewport

`Window.render(time=0.0, frame_time=0.0)` → None

Renders a frame by calling the configured render callback

#### Keyword Arguments

- **time** (*float*) – Current time in seconds
- **frame\_time** (*float*) – Delta time from last frame in seconds

`Window.swap_buffers()`

Swap buffers, increment frame counter and pull events

`Window.resize(width, height)` → None

Should be called every time window is resized so the example can adapt to the new size if needed

`Window.destroy()`

Gracefully terminate GLFW

`Window.set_default_viewport()` → None

Calculates the and sets the viewport based on window configuration.

The viewport will based on the configured fixed aspect ratio if set. If no fixed aspect ratio is set the viewport will be scaled to the entire window size regardless of size.

Will add black borders and center the viewport if the window do not match the configured viewport (fixed only)

`Window.print_context_info()`

Prints moderngl context info.

## 9.3.2 Attributes

`Window.keys`

GLFW specific key constants

`Window.ctx`

The ModernGL context for the window

**Type** moderngl.Context

`Window.fbo`

The default framebuffer

**Type** moderngl.Framebuffer

**Window.title**

Window title.

This property can also be set:

```
window.title = "New Title"
```

**Type** str

**Window.gl\_version**

(major, minor) required OpenGL version

**Type** Tuple[int, int]

**Window.width**

The current window width

**Type** int

**Window.height**

The current window height

**Type** int

**Window.size**

current window size.

This property also support assignment:

```
# Resize the window to 1000 x 1000
window.size = 1000, 1000
```

**Type** Tuple[int, int]

**Window.position**

The current window position.

This property can also be set to move the window:

```
# Move window to 100, 100
window.position = 100, 100
```

**Type** Tuple[int, int]

**Window.buffer\_width**

the current window buffer width

**Type** int

**Window.buffer\_height**

the current window buffer height

**Type** int

**Window.buffer\_size**

tuple with the current window buffer size

**Type** Tuple[int, int]



Window.**pixel\_ratio**

The framebuffer/window size ratio

**Type** float

Window.**viewport**

current window viewport

**Type** Tuple[int, int, int, int]

Window.**viewport\_size**

Size of the viewport.

Equivalent to `self.viewport[2], self.viewport[3]`

**Type** Tuple[int,int]

Window.**viewport\_width**

The width of the viewport.

Equivalent to `self.viewport[2]`.

**Type** int

Window.**viewport\_height**

The height of the viewport

Equivalent to `self.viewport[3]`.

**Type** int

Window.**frames**

Number of frames rendered

**Type** int

Window.**resizable**

Window is resizable

**Type** bool

Window.**close\_func**

Get or set the close callable

**Type** callable

Window.**fullscreen**

Window is in fullscreen mode

**Type** bool

Window.**config**

Get the current WindowConfig instance

This property can also be set. Assigning a WindowConfig instance will automatically set up the necessary event callback methods:

```
window.config = window_config_instance
```

Window.**vsync**

vertical sync enabled/disabled

**Type** bool

**Window.aspect\_ratio**

The current aspect ratio of the window. If a fixed aspect ratio was passed to the window initializer this value will always be returned. Otherwise `width / height` will be returned.

This property is read only.

**Type** float

**Window.fixed\_aspect\_ratio**

The fixed aspect ratio for the window.

Can be set to `None` to disable fixed aspect ratio making the aspect ratio adjust to the actual window size

This will affects how the viewport is calculated and the reported value from the `aspect_ratio` property:

```
# Enabled fixed aspect ratio
window.fixed_aspect_ratio = 16 / 9

# Disable fixed aspect ratio
window.fixed_aspect_ratio = None
```

**Type** float

**Window.samples**

Number of Multisample anti-aliasing (MSAA) samples

**Type** float

**Window.cursor**

Should the mouse cursor be visible inside the window?

This property can also be assigned to:

```
# Disable cursor
window.cursor = False
```

**Type** bool

**Window.mouse\_exclusivity**

If mouse exclusivity is enabled.

When you enable mouse-exclusive mode, the mouse cursor is no longer available. It is not merely hidden – no amount of mouse movement will make it leave your application. This is for example useful when you don't want the mouse leaving the screen when rotating a 3d scene.

This property can also be set:

```
window.mouse_exclusivity = True
```

**Type** bool

**Window.render\_func**

The render callable

This property can also be used to assign a callable.

**Type** callable

**Window.resize\_func**

Get or set the resize callable

**Type** callable

`Window.iconify_func`

Get or set the iconify/show/hide callable

**Type** callable

`Window.key_event_func`

Get or set the key\_event callable

**Type** callable

`Window.mouse_position_event_func`

Get or set the mouse\_position callable

**Type** callable

`Window.mouse_press_event_func`

Get or set the mouse\_press callable

**Type** callable

`Window.mouse_release_event_func`

Get or set the mouse\_release callable

**Type** callable

`Window.mouse_drag_event_func`

Get or set the mouse\_drag callable

**Type** callable

`Window.mouse_scroll_event_func`

Get or set the mouse\_scroll\_event callable

**Type** callable

`Window.unicode_char_entered_func`

Get or set the unicode\_char\_entered callable

**Type** callable

`Window.is_closing`

Checks if the window is scheduled for closing

**Type** bool

`Window.mouse = <class 'moderngl_window.context.base.window.MouseButtons'>`

`Window.mouse_states`

Mouse button state structure.

The current mouse button states.

```
window.mouse_buttons.left
window.mouse_buttons.right
window.mouse_buttons.middle
```

**Type** MouseButtonStates

`Window.modifiers`

(KeyModifiers) The current keyboard modifiers

Window.**gl\_version\_code**

Generates the version code integer for the selected OpenGL version.

gl\_version (4, 1) returns 410

**Type** int

### 9.3.3 Window Specific Methods

Window.**glfw\_window\_resize\_callback** (*window, width, height*)

Window resize callback for glfw

**Parameters**

- **window** – The window
- **width** – New width
- **height** – New height

Window.**glfw\_mouse\_event\_callback** (*window, xpos, ypos*)

Mouse position event callback from glfw. Translates the events forwarding them to `cursor_event()`.

Screen coordinates relative to the top-left corner

**Parameters**

- **window** – The window
- **xpos** – viewport x pos
- **ypos** – viewport y pos

Window.**glfw\_mouse\_button\_callback** (*window, button, action, mods*)

Handle mouse button events and forward them to the example

**Parameters**

- **window** – The window
- **button** – The button creating the event
- **action** – Button action (press or release)
- **mods** – They modifiers such as ctrl or shift

Window.**glfw\_mouse\_scroll\_callback** (*window, x\_offset: float, y\_offset: float*)

Handle mouse scroll events and forward them to the example

**Parameters**

- **window** – The window
- **x\_offset** (*float*) – x wheel offset
- **y\_offset** (*float*) – y wheel offset

Window.**glfw\_key\_event\_callback** (*window, key, scancode, action, mods*)

Key event callback for glfw. Translates and forwards keyboard event to `keyboard_event()`

**Parameters**

- **window** – Window event origin
- **key** – The key that was pressed or released.
- **scancode** – The system-specific scancode of the key.

- **action** – GLFW\_PRESS, GLFW\_RELEASE or GLFW\_REPEAT
- **mods** – Bit field describing which modifier keys were held down.

Window.**glfw\_char\_callback** (*window*, *codepoint*: *int*)

Handle text input (only unicode charaters)

#### Parameters

- **window** – The glfw window
- **codepoint** (*int*) – The unicode codepoint

Window.**glfw\_cursor\_enter** (*window*, *enter*: *int*)

called when the cursor enters or leaves the content area of the window.

#### Parameters

- **window** – the window instance
- **enter** (*int*) – 0: leave, 1: enter

Window.**glfw\_window\_focus** (*window*, *focused*: *int*)

Called when the window focus is changed.

#### Parameters

- **window** – The window instance
- **focused** (*int*) – 0: de-focus, 1: focused

Window.**glfw\_window\_iconify** (*window*, *iconified*: *int*)

Called when the window is minimized or restored.

#### Parameters

- **window** – The window
- **iconified** (*int*) – 1 = minimized, 0 = restored.

## 9.4 headless.Window

### 9.4.1 Methods

Window.**\_\_init\_\_** (*\*\*kwargs*)

Initialize a window instance.

#### Parameters

- **title** (*str*) – The window title
- **gl\_version** (*tuple*) – Major and minor version of the opengl context to create
- **size** (*tuple*) – Window size x, y
- **resizable** (*bool*) – Should the window be resizable?
- **fullscreen** (*bool*) – Open window in fullscreen mode
- **vsync** (*bool*) – Enable/disable vsync
- **aspect\_ratio** (*float*) – The desired fixed aspect ratio. Can be set to `None` to make aspect ratio be based on the actual window size.
- **samples** (*int*) – Number of MSAA samples for the default framebuffer

- **cursor** (*bool*) – Enable/disable displaying the cursor inside the window

`Window.init_mgl_context()` → None

Create an standalone context and framebuffer

`Window.is_key_pressed(key)` → bool

Returns: The press state of a key

`Window.close()` → None

Signal for the window to close

`Window.use()`

Bind the window's framebuffer

`Window.clear(red=0.0, green=0.0, blue=0.0, alpha=0.0, depth=1.0, viewport=None)`

Binds and clears the default framebuffer

#### Parameters

- **red** (*float*) – color component
- **green** (*float*) – color component
- **blue** (*float*) – color component
- **alpha** (*float*) – alpha component
- **depth** (*float*) – depth value
- **viewport** (*tuple*) – The viewport

`Window.render(time=0.0, frame_time=0.0)` → None

Renders a frame by calling the configured render callback

#### Keyword Arguments

- **time** (*float*) – Current time in seconds
- **frame\_time** (*float*) – Delta time from last frame in seconds

`Window.swap_buffers()` → None

Placeholder. We currently don't do double buffering in headless mode. This may change in the future.

`Window.resize(width, height)` → None

Should be called every time window is resized so the example can adapt to the new size if needed

`Window.destroy()` → None

A library specific destroy method is required

`Window.set_default_viewport()` → None

Calculates the and sets the viewport based on window configuration.

The viewport will based on the configured fixed aspect ratio if set. If no fixed aspect ratio is set the viewport will be scaled to the entire window size regardless of size.

Will add black borders and center the viewport if the window do not match the configured viewport (fixed only)

`Window.print_context_info()`

Prints moderngl context info.

## 9.4.2 Attributes

`Window.keys`

Window.**ctx**

The ModernGL context for the window

**Type** moderngl.Context

Window.**fbo**

The default framebuffer

**Type** moderngl.Framebuffer

Window.**title**

Window title.

This property can also be set:

```
window.title = "New Title"
```

**Type** str

Window.**gl\_version**

(major, minor) required OpenGL version

**Type** Tuple[int, int]

Window.**width**

The current window width

**Type** int

Window.**height**

The current window height

**Type** int

Window.**size**

current window size.

This property also support assignment:

```
# Resize the window to 1000 x 1000
window.size = 1000, 1000
```

**Type** Tuple[int, int]

Window.**position**

The current window position.

This property can also be set to move the window:

```
# Move window to 100, 100
window.position = 100, 100
```

**Type** Tuple[int, int]

Window.**buffer\_width**

the current window buffer width

**Type** int

Window.**buffer\_height**

the current window buffer height

**Type** int

Window.**buffer\_size**

tuple with the current window buffer size

**Type** Tuple[int, int]

Window.**pixel\_ratio**

The framebuffer/window size ratio

**Type** float

Window.**viewport**

current window viewport

**Type** Tuple[int, int, int, int]

Window.**viewport\_size**

Size of the viewport.

Equivalent to `self.viewport[2], self.viewport[3]`

**Type** Tuple[int,int]

Window.**viewport\_width**

The width of the viewport.

Equivalent to `self.viewport[2]`.

**Type** int

Window.**viewport\_height**

The height of the viewport

Equivalent to `self.viewport[3]`.

**Type** int

Window.**frames**

Number of frames rendered

**Type** int

Window.**resizable**

Window is resizable

**Type** bool

Window.**fullscreen**

Window is in fullscreen mode

**Type** bool

Window.**config**

Get the current WindowConfig instance

This property can also be set. Assigning a WindowConfig instance will automatically set up the necessary event callback methods:

```
window.config = window_config_instance
```

Window.**vsync**

vertical sync enabled/disabled

**Type** bool



**Window.aspect\_ratio**

The current aspect ratio of the window. If a fixed aspect ratio was passed to the window initializer this value will always be returned. Otherwise `width / height` will be returned.

This property is read only.

**Type** float

**Window.fixed\_aspect\_ratio**

The fixed aspect ratio for the window.

Can be set to `None` to disable fixed aspect ratio making the aspect ratio adjust to the actual window size

This will affects how the viewport is calculated and the reported value from the `aspect_ratio` property:

```
# Enabled fixed aspect ratio
window.fixed_aspect_ratio = 16 / 9

# Disable fixed aspect ratio
window.fixed_aspect_ratio = None
```

**Type** float

**Window.samples**

Number of Multisample anti-aliasing (MSAA) samples

**Type** float

**Window.cursor**

Should the mouse cursor be visible inside the window?

This property can also be assigned to:

```
# Disable cursor
window.cursor = False
```

**Type** bool

**Window.mouse\_exclusivity**

If mouse exclusivity is enabled.

When you enable mouse-exclusive mode, the mouse cursor is no longer available. It is not merely hidden – no amount of mouse movement will make it leave your application. This is for example useful when you don't want the mouse leaving the screen when rotating a 3d scene.

This property can also be set:

```
window.mouse_exclusivity = True
```

**Type** bool

**Window.render\_func**

The render callable

This property can also be used to assign a callable.

**Type** callable

**Window.resize\_func**

Get or set the resize callable

**Type** callable

Window.**close\_func**

Get or set the close callable

**Type** callable

Window.**iconify\_func**

Get or set the iconify/show/hide callable

**Type** callable

Window.**key\_event\_func**

Get or set the key\_event callable

**Type** callable

Window.**mouse\_position\_event\_func**

Get or set the mouse\_position callable

**Type** callable

Window.**mouse\_press\_event\_func**

Get or set the mouse\_press callable

**Type** callable

Window.**mouse\_release\_event\_func**

Get or set the mouse\_release callable

**Type** callable

Window.**mouse\_drag\_event\_func**

Get or set the mouse\_drag callable

**Type** callable

Window.**mouse\_scroll\_event\_func**

Get or set the mouse\_scroll\_event callable

**Type** callable

Window.**unicode\_char\_entered\_func**

Get or set the unicode\_char\_entered callable

**Type** callable

Window.**is\_closing**

Is the window about to close?

**Type** bool

Window.**mouse** = <class 'moderngl\_window.context.base.window.MouseButtons'>

Window.**mouse\_states**

Mouse button state structure.

The current mouse button states.

```
window.mouse_buttons.left  
window.mouse_buttons.right  
window.mouse_buttons.middle
```

**Type** MouseButtonStates

`Window.modifiers`

(KeyModifiers) The current keyboard modifiers

`Window.gl_version_code`

Generates the version code integer for the selected OpenGL version.

`gl_version (4, 1)` returns 410

**Type** int

## 9.5 pyglet.Window

### 9.5.1 Methods

`Window.__init__ (**kwargs)`

Initialize a window instance.

#### Parameters

- **title** (*str*) – The window title
- **gl\_version** (*tuple*) – Major and minor version of the opengl context to create
- **size** (*tuple*) – Window size x, y
- **resizable** (*bool*) – Should the window be resizable?
- **fullscreen** (*bool*) – Open window in fullscreen mode
- **vsync** (*bool*) – Enable/disable vsync
- **aspect\_ratio** (*float*) – The desired fixed aspect ratio. Can be set to `None` to make aspect ratio be based on the actual window size.
- **samples** (*int*) – Number of MSAA samples for the default framebuffer
- **cursor** (*bool*) – Enable/disable displaying the cursor inside the window

`Window.init_mgl_context ()` → `None`

Create or assign a ModernGL context. If no context is supplied a context will be created using the window's `gl_version`.

**Keyword Arguments** **ctx** – An optional custom ModernGL context

`Window.is_key_pressed (key)` → `bool`

Returns: The press state of a key

`Window.close ()` → `None`

Close the pyglet window directly

`Window.use ()`

Bind the window's framebuffer

`Window.clear (red=0.0, green=0.0, blue=0.0, alpha=0.0, depth=1.0, viewport=None)`

Binds and clears the default framebuffer

#### Parameters

- **red** (*float*) – color component
- **green** (*float*) – color component
- **blue** (*float*) – color component

- **alpha** (*float*) – alpha component
- **depth** (*float*) – depth value
- **viewport** (*tuple*) – The viewport

`Window.render (time=0.0, frame_time=0.0) → None`  
 Renders a frame by calling the configured render callback

#### Keyword Arguments

- **time** (*float*) – Current time in seconds
- **frame\_time** (*float*) – Delta time from last frame in seconds

`Window.swap_buffers () → None`  
 Swap buffers, increment frame counter and pull events

`Window.resize (width, height) → None`  
 Should be called every time window is resized so the example can adapt to the new size if needed

`Window.destroy ()`  
 Destroy the pyglet window

`Window.set_default_viewport () → None`  
 Calculates the and sets the viewport based on window configuration.

The viewport will based on the configured fixed aspect ratio if set. If no fixed aspect ratio is set the viewport will be scaled to the entire window size regardless of size.

Will add black borders and center the viewport if the window do not match the configured viewport (fixed only)

`Window.print_context_info ()`  
 Prints moderngl context info.

## 9.5.2 Window Specific Methods

`Window.on_mouse_press (x: int, y: int, button, mods)`  
 Handle mouse press events and forward to standard methods

#### Parameters

- **x** – x position of the mouse when pressed
- **y** – y position of the mouse when pressed
- **button** – The pressed button
- **mods** – Modifiers

`Window.on_key_release (symbol, modifiers)`  
 Pyglet specific key release callback.

Forwards and translates the events to standard methods.

#### Parameters

- **symbol** – The symbol of the pressed key
- **modifiers** – Modifier state (shift, ctrl etc.)

`Window.on_mouse_drag (x, y, dx, dy, buttons, modifiers)`  
 Pyglet specific mouse drag event.

When a mouse button is pressed this is the only way to capture mouse position events

`Window.on_key_press` (*symbol, modifiers*)

Pyglet specific key press callback.

Forwards and translates the events to the standard methods.

**Parameters**

- **symbol** – The symbol of the pressed key
- **modifiers** – Modifier state (shift, ctrl etc.)

`Window.on_mouse_release` (*x: int, y: int, button, mods*)

Handle mouse release events and forward to standard methods

**Parameters**

- **x** – x position when mouse button was released
- **y** – y position when mouse button was released
- **button** – The button pressed
- **mods** – Modifiers

`Window.on_mouse_motion` (*x, y, dx, dy*)

Pyglet specific mouse motion callback.

Forwards and translates the event to the standard methods.

**Parameters**

- **x** – x position of the mouse
- **y** – y position of the mouse
- **dx** – delta x position
- **dy** – delta y position of the mouse

`Window.on_mouse_scroll` (*x, y, x\_offset: float, y\_offset: float*)

Handle mouse wheel.

**Parameters**

- **x\_offset** (*float*) – X scroll offset
- **y\_offset** (*float*) – Y scroll offset

`Window.on_text` (*text*)

Pyglet specific text input callback

Forwards and translates the events to the standard methods.

**Parameters** **text** (*str*) – The unicode character entered

`Window.on_resize` (*width: int, height: int*)

Pyglet specific callback for window resize events forwarding to standard methods

**Parameters**

- **width** – New window width
- **height** – New window height

`Window.on_show` ()

Called when window first appear or restored from hidden state

`Window.on_hide` ()

Called when window is minimized

`Window.on_close()`  
Pyglet specific window close callback

### 9.5.3 Attributes

`Window.keys`  
Pyglet specific key constants

`Window.ctx`  
The ModernGL context for the window  
**Type** `moderngl.Context`

`Window.fbo`  
The default framebuffer  
**Type** `moderngl.Framebuffer`

`Window.title`  
Window title.  
This property can also be set:

```
window.title = "New Title"
```

**Type** `str`

`Window.gl_version`  
(major, minor) required OpenGL version  
**Type** `Tuple[int, int]`

`Window.width`  
The current window width  
**Type** `int`

`Window.height`  
The current window height  
**Type** `int`

`Window.size`  
current window size.  
This property also support assignment:

```
# Resize the window to 1000 x 1000
window.size = 1000, 1000
```

**Type** `Tuple[int, int]`

`Window.position`  
The current window position.

This property can also be set to move the window:

```
# Move window to 100, 100
window.position = 100, 100
```

**Type** Tuple[int, int]

**Window.buffer\_width**  
the current window buffer width

**Type** int

**Window.buffer\_height**  
the current window buffer height

**Type** int

**Window.buffer\_size**  
tuple with the current window buffer size

**Type** Tuple[int, int]

**Window.pixel\_ratio**  
The framebuffer/window size ratio

**Type** float

**Window.viewport**  
current window viewport

**Type** Tuple[int, int, int, int]

**Window.viewport\_size**  
Size of the viewport.  
  
Equivalent to `self.viewport[2], self.viewport[3]`

**Type** Tuple[int,int]

**Window.viewport\_width**  
The width of the viewport.  
  
Equivalent to `self.viewport[2]`.

**Type** int

**Window.viewport\_height**  
The height of the viewport  
  
Equivalent to `self.viewport[3]`.

**Type** int

**Window.frames**  
Number of frames rendered

**Type** int

**Window.resizable**  
Window is resizable

**Type** bool

**Window.fullscreen**  
Window is in fullscreen mode

**Type** bool

**Window.config**  
Get the current WindowConfig instance

This property can also be set. Assigning a WindowConfig instance will automatically set up the necessary event callback methods:

```
window.config = window_config_instance
```

#### Window.**vsync**

vertical sync enabled/disabled

**Type** bool

#### Window.**aspect\_ratio**

The current aspect ratio of the window. If a fixed aspect ratio was passed to the window initializer this value will always be returned. Otherwise width / height will be returned.

This property is read only.

**Type** float

#### Window.**fixed\_aspect\_ratio**

The fixed aspect ratio for the window.

Can be set to None to disable fixed aspect ratio making the aspect ratio adjust to the actual window size

This will affect how the viewport is calculated and the reported value from the aspect\_ratio property:

```
# Enabled fixed aspect ratio
window.fixed_aspect_ratio = 16 / 9

# Disable fixed aspect ratio
window.fixed_aspect_ratio = None
```

**Type** float

#### Window.**samples**

Number of Multisample anti-aliasing (MSAA) samples

**Type** float

#### Window.**cursor**

Should the mouse cursor be visible inside the window?

This property can also be assigned to:

```
# Disable cursor
window.cursor = False
```

**Type** bool

#### Window.**mouse\_exclusivity**

If mouse exclusivity is enabled.

When you enable mouse-exclusive mode, the mouse cursor is no longer available. It is not merely hidden – no amount of mouse movement will make it leave your application. This is for example useful when you don't want the mouse leaving the screen when rotating a 3d scene.

This property can also be set:

```
window.mouse_exclusivity = True
```

**Type** bool



`Window.render_func`

The render callable

This property can also be used to assign a callable.

**Type** callable

`Window.resize_func`

Get or set the resize callable

**Type** callable

`Window.close_func`

Get or set the close callable

**Type** callable

`Window.iconify_func`

Get or set the iconify/show/hide callable

**Type** callable

`Window.key_event_func`

Get or set the key\_event callable

**Type** callable

`Window.mouse_position_event_func`

Get or set the mouse\_position callable

**Type** callable

`Window.mouse_press_event_func`

Get or set the mouse\_press callable

**Type** callable

`Window.mouse_release_event_func`

Get or set the mouse\_release callable

**Type** callable

`Window.mouse_drag_event_func`

Get or set the mouse\_drag callable

**Type** callable

`Window.unicode_char_entered_func`

Get or set the unicode\_char\_entered callable

**Type** callable

`Window.mouse_scroll_event_func`

Get or set the mouse\_scroll\_event callable

**Type** callable

`Window.is_closing`

Check pyglet's internal exit state

`Window.mouse = <class 'moderngl_window.context.base.window.MouseButtons'>`

`Window.mouse_states`

Mouse button state structure.

The current mouse button states.

```
window.mouse_buttons.left
window.mouse_buttons.right
window.mouse_buttons.middle
```

**Type** MouseButtonStates

Window.**modifiers**

(KeyModifiers) The current keyboard modifiers

Window.**gl\_version\_code**

Generates the version code integer for the selected OpenGL version.

gl\_version (4, 1) returns 410

**Type** int

## 9.6 PyQt5.Window

### 9.6.1 Methods

Window.**\_\_init\_\_**(\*\*kwargs)

Initialize a window instance.

**Parameters**

- **title** (*str*) – The window title
- **gl\_version** (*tuple*) – Major and minor version of the opengl context to create
- **size** (*tuple*) – Window size x, y
- **resizable** (*bool*) – Should the window be resizable?
- **fullscreen** (*bool*) – Open window in fullscreen mode
- **vsync** (*bool*) – Enable/disable vsync
- **aspect\_ratio** (*float*) – The desired fixed aspect ratio. Can be set to None to make aspect ratio be based on the actual window size.
- **samples** (*int*) – Number of MSAA samples for the default framebuffer
- **cursor** (*bool*) – Enable/disable displaying the cursor inside the window

Window.**init\_mgl\_context**() → None

Create or assign a ModernGL context. If no context is supplied a context will be created using the window's gl\_version.

**Keyword Arguments** **ctx** – An optional custom ModernGL context

Window.**is\_key\_pressed**(key) → bool

Returns: The press state of a key

Window.**close**()

Close the window

Window.**use**()

Bind the window's framebuffer

Window.**clear**(red=0.0, green=0.0, blue=0.0, alpha=0.0, depth=1.0, viewport=None)

Binds and clears the default framebuffer

**Parameters**

- **red** (*float*) – color component
- **green** (*float*) – color component
- **blue** (*float*) – color component
- **alpha** (*float*) – alpha component
- **depth** (*float*) – depth value
- **viewport** (*tuple*) – The viewport

Window.**render** (*time=0.0, frame\_time=0.0*) → None  
 Renders a frame by calling the configured render callback

**Keyword Arguments**

- **time** (*float*) – Current time in seconds
- **frame\_time** (*float*) – Delta time from last frame in seconds

Window.**swap\_buffers** () → None  
 Swap buffers, set viewport, trigger events and increment frame counter

Window.**resize** (*width: int, height: int*) → None  
 Replacement for Qt's `resizeGL` method.

**Parameters**

- **width** – New window width
- **height** – New window height

Window.**destroy** () → None  
 Quit the Qt application to exit the window gracefully

Window.**set\_default\_viewport** () → None  
 Calculates the and sets the viewport based on window configuration.

The viewport will based on the configured fixed aspect ratio if set. If no fixed aspect ratio is set the viewport will be scaled to the entire window size regardless of size.

Will add black borders and center the viewport if the window do not match the configured viewport (fixed only)

Window.**print\_context\_info** ()  
 Prints moderngl context info.

## 9.6.2 Window Specific Methods

Window.**close\_event** (*event*) → None  
 The standard PyQt close events

**Parameters** **event** – The qtevent instance

Window.**mouse\_release\_event** (*event*) → None  
 Forward mouse release events to standard methods

**Parameters** **event** – The qtevent instance

Window.**key\_release\_event** (*event*) → None  
 Process Qt key release events forwarding them to standard methods

**Parameters** **event** – The qtevent instance

Window.**mouse\_move\_event** (*event*) → None

Forward mouse cursor position events to standard methods

**Parameters** **event** – The qtevent instance

Window.**key\_pressed\_event** (*event*) → None

Process Qt key press events forwarding them to standard methods

**Parameters** **event** – The qtevent instance

Window.**mouse\_press\_event** (*event*) → None

Forward mouse press events to standard methods

**Parameters** **event** – The qtevent instance

Window.**mouse\_wheel\_event** (*event*)

Forward mouse wheel events to standard methods.

From Qt docs:

Returns the distance that the wheel is rotated, in eighths of a degree. A positive value indicates that the wheel was rotated forwards away from the user; a negative value indicates that the wheel was rotated backwards toward the user.

Most mouse types work in steps of 15 degrees, in which case the delta value is a multiple of 120; i.e., 120 units \* 1/8 = 15 degrees.

However, some mice have finer-resolution wheels and send delta values that are less than 120 units (less than 15 degrees). To support this possibility, you can either cumulatively add the delta values from events until the value of 120 is reached, then scroll the widget, or you can partially scroll the widget in response to each wheel event.

**Parameters** **event** (*QWheelEvent*) – Mouse wheel event

Window.**show\_event** (*event*)

The standard Qt show event

Window.**hide\_event** (*event*)

The standard Qt hide event

### 9.6.3 Attributes

Window.**keys**

PyQt5 specific key constants

Window.**ctx**

The ModernGL context for the window

**Type** moderngl.Context

Window.**fbo**

The default framebuffer

**Type** moderngl.Framebuffer

Window.**title**

Window title.

This property can also be set:

```
window.title = "New Title"
```

**Type** str

Window.**gl\_version**

(major, minor) required OpenGL version

**Type** Tuple[int, int]

Window.**width**

The current window width

**Type** int

Window.**height**

The current window height

**Type** int

Window.**size**

current window size.

This property also support assignment:

```
# Resize the window to 1000 x 1000
window.size = 1000, 1000
```

**Type** Tuple[int, int]

Window.**position**

The current window position.

This property can also be set to move the window:

```
# Move window to 100, 100
window.position = 100, 100
```

**Type** Tuple[int, int]

Window.**buffer\_width**

the current window buffer width

**Type** int

Window.**buffer\_height**

the current window buffer height

**Type** int

Window.**buffer\_size**

tuple with the current window buffer size

**Type** Tuple[int, int]

Window.**pixel\_ratio**

The framebuffer/window size ratio

**Type** float

Window.**viewport**

current window viewport

**Type** Tuple[int, int, int, int]

**Window.viewport\_size**

Size of the viewport.

Equivalent to `self.viewport[2], self.viewport[3]`

**Type** `Tuple[int,int]`

**Window.viewport\_width**

The width of the viewport.

Equivalent to `self.viewport[2]`.

**Type** `int`

**Window.viewport\_height**

The height of the viewport

Equivalent to `self.viewport[3]`.

**Type** `int`

**Window.frames**

Number of frames rendered

**Type** `int`

**Window.resizable**

Window is resizable

**Type** `bool`

**Window.fullscreen**

Window is in fullscreen mode

**Type** `bool`

**Window.config**

Get the current `WindowConfig` instance

This property can also be set. Assigning a `WindowConfig` instance will automatically set up the necessary event callback methods:

```
window.config = window_config_instance
```

**Window.vsync**

vertical sync enabled/disabled

**Type** `bool`

**Window.aspect\_ratio**

The current aspect ratio of the window. If a fixed aspect ratio was passed to the window initializer this value will always be returned. Otherwise `width / height` will be returned.

This property is read only.

**Type** `float`

**Window.fixed\_aspect\_ratio**

The fixed aspect ratio for the window.

Can be set to `None` to disable fixed aspect ratio making the aspect ratio adjust to the actual window size

This will affects how the viewport is calculated and the reported value from the `aspect_ratio` property:

```
# Enabled fixed aspect ratio
window.fixed_aspect_ratio = 16 / 9

# Disable fixed aspect ratio
window.fixed_aspect_ratio = None
```

**Type** float

#### Window.**samples**

Number of Multisample anti-aliasing (MSAA) samples

**Type** float

#### Window.**cursor**

Should the mouse cursor be visible inside the window?

This property can also be assigned to:

```
# Disable cursor
window.cursor = False
```

**Type** bool

#### Window.**mouse\_exclusivity**

If mouse exclusivity is enabled.

When you enable mouse-exclusive mode, the mouse cursor is no longer available. It is not merely hidden – no amount of mouse movement will make it leave your application. This is for example useful when you don't want the mouse leaving the screen when rotating a 3d scene.

This property can also be set:

```
window.mouse_exclusivity = True
```

**Type** bool

#### Window.**render\_func**

The render callable

This property can also be used to assign a callable.

**Type** callable

#### Window.**resize\_func**

Get or set the resize callable

**Type** callable

#### Window.**close\_func**

Get or set the close callable

**Type** callable

#### Window.**iconify\_func**

Get or set the iconify/show/hide callable

**Type** callable

#### Window.**key\_event\_func**

Get or set the key\_event callable

**Type** callable

`Window.mouse_position_event_func`

Get or set the mouse\_position callable

**Type** callable

`Window.mouse_press_event_func`

Get or set the mouse\_press callable

**Type** callable

`Window.mouse_release_event_func`

Get or set the mouse\_release callable

**Type** callable

`Window.mouse_drag_event_func`

Get or set the mouse\_drag callable

**Type** callable

`Window.unicode_char_entered_func`

Get or set the unicode\_char\_entered callable

**Type** callable

`Window.mouse_scroll_event_func`

Get or set the mouse\_scroll\_event callable

**Type** callable

`Window.is_closing`

Is the window about to close?

**Type** bool

`Window.mouse = <class 'moderngl_window.context.base.window.MouseButtons'>`

`Window.mouse_states`

Mouse button state structure.

The current mouse button states.

```
window.mouse_buttons.left
window.mouse_buttons.right
window.mouse_buttons.middle
```

**Type** MouseButtonStates

`Window.modifiers`

(KeyModifiers) The current keyboard modifiers

`Window.gl_version_code`

Generates the version code integer for the selected OpenGL version.

gl\_version (4, 1) returns 410

**Type** int



## 9.7 pyside2.Window

### 9.7.1 Methods

`Window.__init__ (**kwargs)`

Initialize a window instance.

#### Parameters

- **title** (*str*) – The window title
- **gl\_version** (*tuple*) – Major and minor version of the opengl context to create
- **size** (*tuple*) – Window size x, y
- **resizable** (*bool*) – Should the window be resizable?
- **fullscreen** (*bool*) – Open window in fullscreen mode
- **vsync** (*bool*) – Enable/disable vsync
- **aspect\_ratio** (*float*) – The desired fixed aspect ratio. Can be set to `None` to make aspect ratio be based on the actual window size.
- **samples** (*int*) – Number of MSAA samples for the default framebuffer
- **cursor** (*bool*) – Enable/disable displaying the cursor inside the window

`Window.init_mgl_context ()` → `None`

Create or assign a ModernGL context. If no context is supplied a context will be created using the window's `gl_version`.

**Keyword Arguments** `ctx` – An optional custom ModernGL context

`Window.is_key_pressed (key)` → `bool`

Returns: The press state of a key

`Window.close ()`

Close the window

`Window.use ()`

Bind the window's framebuffer

`Window.clear (red=0.0, green=0.0, blue=0.0, alpha=0.0, depth=1.0, viewport=None)`

Binds and clears the default framebuffer

#### Parameters

- **red** (*float*) – color component
- **green** (*float*) – color component
- **blue** (*float*) – color component
- **alpha** (*float*) – alpha component
- **depth** (*float*) – depth value
- **viewport** (*tuple*) – The viewport

`Window.render (time=0.0, frame_time=0.0)` → `None`

Renders a frame by calling the configured render callback

#### Keyword Arguments

- **time** (*float*) – Current time in seconds

- **frame\_time** (*float*) – Delta time from last frame in seconds

Window.**swap\_buffers**() → None

Swap buffers, set viewport, trigger events and increment frame counter

Window.**resize**(*width: int, height: int*) → None

Replacement for Qt's `resizeGL` method.

#### Parameters

- **width** – New window width
- **height** – New window height

Window.**destroy**() → None

Quit the Qt application to exit the window gracefully

Window.**set\_default\_viewport**() → None

Calculates the and sets the viewport based on window configuration.

The viewport will based on the configured fixed aspect ratio if set. If no fixed aspect ratio is set the viewport will be scaled to the entire window size regardless of size.

Will add black borders and center the viewport if the window do not match the configured viewport (fixed only)

Window.**print\_context\_info**()

Prints moderngl context info.

## 9.7.2 Window Specific Methods

Window.**close\_event**(*event*) → None

The standard PyQt close events

**Parameters** **event** – The qtevent instance

Window.**mouse\_release\_event**(*event*) → None

Forward mouse release events to standard methods

**Parameters** **event** – The qtevent instance

Window.**key\_release\_event**(*event*)

Process Qt key release events forwarding them to standard methods

**Parameters** **event** – The qtevent instance

Window.**mouse\_move\_event**(*event*) → None

Forward mouse cursor position events to standard methods

**Parameters** **event** – The qtevent instance

Window.**key\_pressed\_event**(*event*)

Process Qt key press events forwarding them to standard methods

**Parameters** **event** – The qtevent instance

Window.**mouse\_press\_event**(*event*) → None

Forward mouse press events to standard methods

**Parameters** **event** – The qtevent instance

Window.**mouse\_wheel\_event**(*event*)

Forward mouse wheel events to standard methods.

From Qt docs:

Returns the distance that the wheel is rotated, in eighths of a degree. A positive value indicates that the wheel was rotated forwards away from the user; a negative value indicates that the wheel was rotated backwards toward the user.

Most mouse types work in steps of 15 degrees, in which case the delta value is a multiple of 120; i.e., 120 units \* 1/8 = 15 degrees.

However, some mice have finer-resolution wheels and send delta values that are less than 120 units (less than 15 degrees). To support this possibility, you can either cumulatively add the delta values from events until the value of 120 is reached, then scroll the widget, or you can partially scroll the widget in response to each wheel event.

**Parameters** `event` (*QWheelEvent*) – Mouse wheel event

`Window.show_event(event)`  
The standard Qt show event

`Window.hide_event(event)`  
The standard Qt hide event

### 9.7.3 Attributes

`Window.keys`  
PySide2 specific key constants

`Window.ctx`  
The ModernGL context for the window

**Type** `moderngl.Context`

`Window.fbo`  
The default framebuffer

**Type** `moderngl.Framebuffer`

`Window.title`  
Window title.

This property can also be set:

```
window.title = "New Title"
```

**Type** `str`

`Window.gl_version`  
(major, minor) required OpenGL version

**Type** `Tuple[int, int]`

`Window.width`  
The current window width

**Type** `int`

`Window.height`  
The current window height

**Type** `int`

**Window.size**

current window size.

This property also support assignment:

```
# Resize the window to 1000 x 1000
window.size = 1000, 1000
```

**Type** Tuple[int, int]

**Window.position**

The current window position.

This property can also be set to move the window:

```
# Move window to 100, 100
window.position = 100, 100
```

**Type** Tuple[int, int]

**Window.buffer\_width**

the current window buffer width

**Type** int

**Window.buffer\_height**

the current window buffer height

**Type** int

**Window.buffer\_size**

tuple with the current window buffer size

**Type** Tuple[int, int]

**Window.pixel\_ratio**

The framebuffer/window size ratio

**Type** float

**Window.viewport**

current window viewport

**Type** Tuple[int, int, int, int]

**Window.viewport\_size**

Size of the viewport.

Equivalent to `self.viewport[2], self.viewport[3]`

**Type** Tuple[int,int]

**Window.viewport\_width**

The width of the viewport.

Equivalent to `self.viewport[2].`

**Type** int

**Window.viewport\_height**

The height of the viewport

Equivalent to `self.viewport[3].`

**Type** int

Window.**frames**

Number of frames rendered

**Type** int

Window.**resizable**

Window is resizable

**Type** bool

Window.**fullscreen**

Window is in fullscreen mode

**Type** bool

Window.**config**

Get the current WindowConfig instance

This property can also be set. Assigning a WindowConfig instance will automatically set up the necessary event callback methods:

```
window.config = window_config_instance
```

Window.**vsync**

vertical sync enabled/disabled

**Type** bool

Window.**aspect\_ratio**

The current aspect ratio of the window. If a fixed aspect ratio was passed to the window initializer this value will always be returned. Otherwise width / height will be returned.

This property is read only.

**Type** float

Window.**fixed\_aspect\_ratio**

The fixed aspect ratio for the window.

Can be set to None to disable fixed aspect ratio making the aspect ratio adjust to the actual window size

This will affects how the viewport is calculated and the reported value from the `aspect_ratio` property:

```
# Enabled fixed aspect ratio
window.fixed_aspect_ratio = 16 / 9

# Disable fixed aspect ratio
window.fixed_aspect_ratio = None
```

**Type** float

Window.**samples**

Number of Multisample anti-aliasing (MSAA) samples

**Type** float

Window.**cursor**

Should the mouse cursor be visible inside the window?

This property can also be assigned to:

```
# Disable cursor
window.cursor = False
```

**Type** bool

**Window.mouse\_exclusivity**

If mouse exclusivity is enabled.

When you enable mouse-exclusive mode, the mouse cursor is no longer available. It is not merely hidden – no amount of mouse movement will make it leave your application. This is for example useful when you don't want the mouse leaving the screen when rotating a 3d scene.

This property can also be set:

```
window.mouse_exclusivity = True
```

**Type** bool

**Window.render\_func**

The render callable

This property can also be used to assign a callable.

**Type** callable

**Window.resize\_func**

Get or set the resize callable

**Type** callable

**Window.close\_func**

Get or set the close callable

**Type** callable

**Window.iconify\_func**

Get or set the iconify/show/hide callable

**Type** callable

**Window.key\_event\_func**

Get or set the key\_event callable

**Type** callable

**Window.mouse\_position\_event\_func**

Get or set the mouse\_position callable

**Type** callable

**Window.mouse\_press\_event\_func**

Get or set the mouse\_press callable

**Type** callable

**Window.mouse\_release\_event\_func**

Get or set the mouse\_release callable

**Type** callable

**Window.mouse\_drag\_event\_func**

Get or set the mouse\_drag callable

**Type** callable

Window.**unicode\_char\_entered\_func**

Get or set the unicode\_char\_entered callable

**Type** callable

Window.**mouse\_scroll\_event\_func**

Get or set the mouse\_scroll\_event callable

**Type** callable

Window.**is\_closing**

Is the window about to close?

**Type** bool

Window.**mouse** = <class 'moderngl\_window.context.base.window.MouseButtons'>

Window.**mouse\_states**

Mouse button state structure.

The current mouse button states.

```

window.mouse_buttons.left
window.mouse_buttons.right
window.mouse_buttons.middle

```

**Type** MouseButtonStates

Window.**modifiers**

(KeyModifiers) The current keyboard modifiers

Window.**gl\_version\_code**

Generates the version code integer for the selected OpenGL version.

gl\_version (4, 1) returns 410

**Type** int

## 9.8 sdl2.Window

### 9.8.1 Methods

Window.**\_\_init\_\_** (\*\*kwargs)

Initialize a window instance.

**Parameters**

- **title** (*str*) – The window title
- **gl\_version** (*tuple*) – Major and minor version of the opengl context to create
- **size** (*tuple*) – Window size x, y
- **resizable** (*bool*) – Should the window be resizable?
- **fullscreen** (*bool*) – Open window in fullscreen mode
- **vsync** (*bool*) – Enable/disable vsync

- **aspect\_ratio** (*float*) – The desired fixed aspect ratio. Can be set to `None` to make aspect ratio be based on the actual window size.
- **samples** (*int*) – Number of MSAA samples for the default framebuffer
- **cursor** (*bool*) – Enable/disable displaying the cursor inside the window

`Window.init_mgl_context()` → `None`

Create or assign a ModernGL context. If no context is supplied a context will be created using the window's `gl_version`.

**Keyword Arguments** `ctx` – An optional custom ModernGL context

`Window.is_key_pressed(key)` → `bool`

Returns: The press state of a key

`Window.close()`

Close the window

`Window.use()`

Bind the window's framebuffer

`Window.clear(red=0.0, green=0.0, blue=0.0, alpha=0.0, depth=1.0, viewport=None)`

Binds and clears the default framebuffer

#### Parameters

- **red** (*float*) – color component
- **green** (*float*) – color component
- **blue** (*float*) – color component
- **alpha** (*float*) – alpha component
- **depth** (*float*) – depth value
- **viewport** (*tuple*) – The viewport

`Window.render(time=0.0, frame_time=0.0)` → `None`

Renders a frame by calling the configured render callback

#### Keyword Arguments

- **time** (*float*) – Current time in seconds
- **frame\_time** (*float*) – Delta time from last frame in seconds

`Window.swap_buffers()` → `None`

Swap buffers, set viewport, trigger events and increment frame counter

`Window.resize(width, height)` → `None`

Resize callback.

#### Parameters

- **width** – New window width
- **height** – New window height

`Window.destroy()` → `None`

Gracefully close the window

`Window.set_default_viewport()` → `None`

Calculates the and sets the viewport based on window configuration.



The viewport will based on the configured fixed aspect ratio if set. If no fixed aspect ratio is set the viewport will be scaled to the entire window size regardless of size.

Will add black borders and center the viewport if the window do not match the configured viewport (fixed only)

`Window.print_context_info()`  
Prints moderngl context info.

## 9.8.2 Window Specific Methods

`Window.process_events()` → None  
Handle all queued events in sdl2 dispatching events to standard methods

## 9.8.3 Attributes

`Window.keys`  
SDL2 specific key constants

`Window.ctx`  
The ModernGL context for the window  
**Type** moderngl.Context

`Window.fbo`  
The default framebuffer  
**Type** moderngl.Framebuffer

`Window.title`  
Window title.  
This property can also be set:

```
window.title = "New Title"
```

**Type** str

`Window.gl_version`  
(major, minor) required OpenGL version  
**Type** Tuple[int, int]

`Window.width`  
The current window width  
**Type** int

`Window.height`  
The current window height  
**Type** int

`Window.size`  
current window size.  
This property also support assignment:

```
# Resize the window to 1000 x 1000
window.size = 1000, 1000
```

**Type** Tuple[int, int]

Window.**position**

The current window position.

This property can also be set to move the window:

```
# Move window to 100, 100
window.position = 100, 100
```

**Type** Tuple[int, int]

Window.**buffer\_width**

the current window buffer width

**Type** int

Window.**buffer\_height**

the current window buffer height

**Type** int

Window.**buffer\_size**

tuple with the current window buffer size

**Type** Tuple[int, int]

Window.**pixel\_ratio**

The framebuffer/window size ratio

**Type** float

Window.**viewport**

current window viewport

**Type** Tuple[int, int, int, int]

Window.**viewport\_size**

Size of the viewport.

Equivalent to `self.viewport[2], self.viewport[3]`

**Type** Tuple[int,int]

Window.**viewport\_width**

The width of the viewport.

Equivalent to `self.viewport[2]`.

**Type** int

Window.**viewport\_height**

The height of the viewport

Equivalent to `self.viewport[3]`.

**Type** int

Window.**frames**

Number of frames rendered

**Type** int

Window.**resizable**

Window is resizable

**Type** bool

`Window.fullscreen`

Window is in fullscreen mode

**Type** bool

`Window.config`

Get the current WindowConfig instance

This property can also be set. Assigning a WindowConfig instance will automatically set up the necessary event callback methods:

```
window.config = window_config_instance
```

`Window.vsync`

vertical sync enabled/disabled

**Type** bool

`Window.aspect_ratio`

The current aspect ratio of the window. If a fixed aspect ratio was passed to the window initializer this value will always be returned. Otherwise `width / height` will be returned.

This property is read only.

**Type** float

`Window.fixed_aspect_ratio`

The fixed aspect ratio for the window.

Can be set to `None` to disable fixed aspect ratio making the aspect ratio adjust to the actual window size

This will affects how the viewport is calculated and the reported value from the `aspect_ratio` property:

```
# Enabled fixed aspect ratio
window.fixed_aspect_ratio = 16 / 9

# Disable fixed aspect ratio
window.fixed_aspect_ratio = None
```

**Type** float

`Window.samples`

Number of Multisample anti-aliasing (MSAA) samples

**Type** float

`Window.cursor`

Should the mouse cursor be visible inside the window?

This property can also be assigned to:

```
# Disable cursor
window.cursor = False
```

**Type** bool

`Window.mouse_exclusivity`

If mouse exclusivity is enabled.

When you enable mouse-exclusive mode, the mouse cursor is no longer available. It is not merely hidden – no amount of mouse movement will make it leave your application. This is for example useful when you don't want the mouse leaving the screen when rotating a 3d scene.

This property can also be set:

```
window.mouse_exclusivity = True
```

**Type** bool

Window.**render\_func**

The render callable

This property can also be used to assign a callable.

**Type** callable

Window.**resize\_func**

Get or set the resize callable

**Type** callable

Window.**close\_func**

Get or set the close callable

**Type** callable

Window.**iconify\_func**

Get or set the iconify/show/hide callable

**Type** callable

Window.**key\_event\_func**

Get or set the key\_event callable

**Type** callable

Window.**mouse\_position\_event\_func**

Get or set the mouse\_position callable

**Type** callable

Window.**mouse\_press\_event\_func**

Get or set the mouse\_press callable

**Type** callable

Window.**mouse\_release\_event\_func**

Get or set the mouse\_release callable

**Type** callable

Window.**mouse\_drag\_event\_func**

Get or set the mouse\_drag callable

**Type** callable

Window.**unicode\_char\_entered\_func**

Get or set the unicode\_char\_entered callable

**Type** callable

Window.**mouse\_scroll\_event\_func**

Get or set the mouse\_scroll\_event callable

**Type** callable

`Window.is_closing`

Is the window about to close?

**Type** bool

`Window.mouse = <class 'moderngl_window.context.base.window.MouseButtons'>`

`Window.mouse_states`

Mouse button state structure.

The current mouse button states.

```
window.mouse_buttons.left  
window.mouse_buttons.right  
window.mouse_buttons.middle
```

**Type** MouseButtonStates

`Window.modifiers`

(KeyModifiers) The current keyboard modifiers

`Window.gl_version_code`

Generates the version code integer for the selected OpenGL version.

`gl_version (4, 1)` returns 410

**Type** int



## MODERNGL\_WINDOW.GEOMETRY

`moderngl_window.geometry.bbox` (*size*=(1.0, 1.0, 1.0), *name*=None, *attr\_names*=<class 'moderngl\_window.geometry.attributes.AttributeNames'>)

Generates a bounding box with (0.0, 0.0, 0.0) as the center. This is simply a box with `LINE_STRIP` as draw mode.

### Keyword Arguments

- **size** (*tuple*) – x, y, z size of the box
- **name** (*str*) – Optional name for the VAO
- **attr\_names** (*AttributeNames*) – Attribute names

**Returns** A `moderngl_window.opengl.vao.VAO` instance

`moderngl_window.geometry.quad_fs` (*attr\_names*=<class 'moderngl\_window.geometry.attributes.AttributeNames'>, *normals*=True, *uvs*=True, *name*=None) → `moderngl_window.opengl.vao.VAO`

Creates a screen aligned quad using two triangles with normals and texture coordinates.

### Keyword Arguments

- **attr\_names** (*AttributeNames*) – Attrib name config
- **normals** (*bool*) – Include normals in VAO
- **uvs** (*bool*) – Include normals in VAO
- **name** (*str*) – Optional name for the VAO

**Returns** A `VAO` instance.

`moderngl_window.geometry.quad_2d` (*size*=(1.0, 1.0), *pos*=(0.0, 0.0), *normals*=True, *uvs*=True, *attr\_names*=<class 'moderngl\_window.geometry.attributes.AttributeNames'>, *name*=None) → `moderngl_window.opengl.vao.VAO`

Creates a 2D quad VAO using 2 triangles with normals and texture coordinates.

### Keyword Arguments

- **size** (*tuple*) – width and height
- **pos** (*float*) – Center position x and y
- **normals** (*bool*) – Include normals in VAO
- **uvs** (*bool*) – Include normals in VAO
- **attr\_names** (*AttributeNames*) – Attrib name config
- **name** (*str*) – Optional name for the VAO

**Returns** A *VAO* instance.

```
moderngl_window.geometry.cube(size=(1.0, 1.0, 1.0), center=(0.0, 0.0, 0.0), normals=True,
                               uvs=True, name=None, attr_names=<class 'mod-
                               erngl_window.geometry.attributes.AttributeNames'>) → mod-
                               erngl_window.opengl.vao.VAO
```

Creates a cube VAO with normals and texture coordinates

#### Keyword Arguments

- **width** (*float*) – Width of the cube
- **height** (*float*) – Height of the cube
- **depth** (*float*) – Depth of the cube
- **center** – center of the cube as a 3-component tuple
- **normals** – (bool) Include normals
- **uvs** – (bool) include uv coordinates
- **name** (*str*) – Optional name for the VAO
- **attr\_names** (*AttributeNames*) – Attribute names

**Returns** A *moderngl\_window.opengl.vao.VAO* instance

```
moderngl_window.geometry.sphere(radius=0.5, sectors=32, rings=16, normals=True,
                                  uvs=True, name: str = None, attr_names=<class 'mod-
                                  erngl_window.geometry.attributes.AttributeNames'>) →
                                  moderngl_window.opengl.vao.VAO
```

Creates a sphere.

#### Keyword Arguments

- **radius** (*float*) – Radius of the sphere
- **rings** (*int*) – number of horizontal rings
- **sectors** (*int*) – number of vertical segments
- **normals** (*bool*) – Include normals in the VAO
- **uvs** (*bool*) – Include texture coordinates in the VAO
- **name** (*str*) – An optional name for the VAO
- **attr\_names** (*AttributeNames*) – Attribute names

**Returns** A VAO instance



## MODERNGL\_WINDOW.LOADERS

### 11.1 base.BaseLoader

#### 11.1.1 Method

`BaseLoader.__init__(meta)`  
Initialize loader.

Loaders take a `ResourceDescription` instance containing all the parameters needed to load and initialize this data.

**Parameters** `meta` (*ResourceDescription*) – The resource to load

**classmethod** `BaseLoader.supports_file(meta)`  
Check if the loader has a supported file extension.

What extensions are supported can be defined in the `file_extensions` class attribute.

`BaseLoader.load()` → Any  
Loads a resource.

When creating a loader this is the only method that needs to be implemented.

**Returns** The loaded resource

`BaseLoader.find_data(path)`  
Find resource using data finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

`BaseLoader.find_program(path)`  
Find resource using program finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

`BaseLoader.find_texture(path)`  
Find resource using texture finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

`BaseLoader.find_scene(path)`  
Find resource using scene finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

### 11.1.2 Attributes

`BaseLoader.kind = None`

The kind of resource this loader supports. This can be used when file extensions is not enough to decide what loader should be selected.

`BaseLoader.file_extensions = []`

A list defining the file extensions accepted by this loader.

Example:

```
# Loader will match .xyz and .xyz.gz files.
file_extensions = [
    ['.xyz'],
    ['.xyz', '.gz'],
]
```

`BaseLoader.ctx`

ModernGL context

**Type** `moderngl.Context`

## 11.2 texture.t2d.Loader

### 11.2.1 Method

`Loader.__init__(meta)`

Initialize loader.

Loaders take a `ResourceDescription` instance containing all the parameters needed to load and initialize this data.

**Parameters** `meta` (*ResourceDescription*) – The resource to load

**classmethod** `Loader.supports_file(meta)`

Check if the loader has a supported file extension.

What extensions are supported can be defined in the `file_extensions` class attribute.

`Loader.load()`

Load a 2d texture as configured in the supplied `TextureDescription`

**Returns** The Texture instance

**Return type** `moderngl.Texture`

`Loader.find_data(path)`

Find resource using data finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

`Loader.find_program(path)`

Find resource using program finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

`Loader.find_texture(path)`

Find resource using texture finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

`Loader.find_scene(path)`

Find resource using scene finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

## 11.2.2 Attributes

`Loader.kind = '2d'`

`Loader.file_extensions = []`

`Loader.ctx`

ModernGL context

**Type** `moderngl.Context`

## 11.3 program.single.Loader

### 11.3.1 Method

`Loader.__init__(meta)`

Initialize loader.

Loaders take a `ResourceDescription` instance containing all the parameters needed to load and initialize this data.

**Parameters** `meta (ResourceDescription)` – The resource to load

**classmethod** `Loader.supports_file(meta)`

Check if the loader has a supported file extension.

What extensions are supported can be defined in the `file_extensions` class attribute.

`Loader.load()` → `moderngl.program.Program`

Loads a shader program from a single glsl file.

Each shader type is separated by preprocessors

- `VERTEX_SHADER`
- `FRAGMENT_SHADER`
- `GEOMETRY_SHADER`
- `TESS_CONTROL_SHADER`
- `TESS_EVALUATION_SHADER`

Example:

```
#version 330

#ifdef VERTEX_SHADER

in vec3 in_position;
in vec2 in_texcoord_0;
out vec2 uv0;

void main() {
    gl_Position = vec4(in_position, 1);
    uv0 = in_texcoord_0;
}

#elif defined FRAGMENT_SHADER

out vec4 fragColor;
uniform sampler2D texture0;
in vec2 uv0;

void main() {
    fragColor = texture(texture0, uv0);
}

#endif
```

**Returns** The Program instance

**Return type** moderngl.Program

Loader.**find\_data** (*path*)

Find resource using data finders.

This is mainly a shortcut method to simplify the task.

**Parameters** *path* – Path to resource

Loader.**find\_program** (*path*)

Find resource using program finders.

This is mainly a shortcut method to simplify the task.

**Parameters** *path* – Path to resource

Loader.**find\_texture** (*path*)

Find resource using texture finders.

This is mainly a shortcut method to simplify the task.

**Parameters** *path* – Path to resource

Loader.**find\_scene** (*path*)

Find resource using scene finders.

This is mainly a shortcut method to simplify the task.

**Parameters** *path* – Path to resource

## 11.3.2 Attributes

Loader.**kind** = 'single'

```
Loader.file_extensions = []
```

```
Loader.ctx
```

ModernGL context

**Type** moderngl.Context

## 11.4 program.separate.Loader

### 11.4.1 Method

```
Loader.__init__(meta)
```

Initialize loader.

Loaders take a ResourceDescription instance containing all the parameters needed to load and initialize this data.

**Parameters** *meta* (*ResourceDescription*) – The resource to load

```
classmethod Loader.supports_file(meta)
```

Check if the loader has a supported file extension.

What extensions are supported can be defined in the *file\_extensions* class attribute.

```
Loader.load() → moderngl.program.Program
```

Loads a shader program where each shader is a separate file.

This detected and dictated by the *kind* in the *ProgramDescription*.

**Returns** The Program instance

**Return type** moderngl.Program

```
Loader.find_data(path)
```

Find resource using data finders.

This is mainly a shortcut method to simplify the task.

**Parameters** *path* – Path to resource

```
Loader.find_program(path)
```

Find resource using program finders.

This is mainly a shortcut method to simplify the task.

**Parameters** *path* – Path to resource

```
Loader.find_texture(path)
```

Find resource using texture finders.

This is mainly a shortcut method to simplify the task.

**Parameters** *path* – Path to resource

```
Loader.find_scene(path)
```

Find resource using scene finders.

This is mainly a shortcut method to simplify the task.

**Parameters** *path* – Path to resource

## 11.4.2 Loader Specific Methods

`Loader.load_shader(shader_type: str, path: str)`  
Load a single shader

## 11.4.3 Attributes

`Loader.kind = 'separate'`

`Loader.file_extensions = []`

`Loader.ctx`

ModernGL context

**Type** moderngl.Context

## 11.5 texture.array.Loader

### 11.5.1 Method

`Loader.__init__(meta)`

Initialize loader.

Loaders take a ResourceDescription instance containing all the parameters needed to load and initialize this data.

**Parameters** `meta` (*ResourceDescription*) – The resource to load

**classmethod** `Loader.supports_file(meta)`

Check if the loader has a supported file extension.

What extensions are supported can be defined in the `file_extensions` class attribute.

`Loader.load()`

Load a texture array as described by the supplied TextureDescription`

**Returns** The TextureArray instance

**Return type** moderngl.TextureArray

`Loader.find_data(path)`

Find resource using data finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

`Loader.find_program(path)`

Find resource using program finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

`Loader.find_texture(path)`

Find resource using texture finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

`Loader.find_scene(path)`

Find resource using scene finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

## 11.5.2 Attributes

`Loader.kind = 'array'`

`Loader.file_extensions = []`

`Loader.ctx`

ModernGL context

**Type** `moderngl.Context`

## 11.6 scene.wavefront.Loader

### 11.6.1 Method

`Loader.__init__(meta: moderngl_window.meta.scene.SceneDescription)`

Initialize loader.

Loaders take a `ResourceDescription` instance containing all the parameters needed to load and initialize this data.

**Parameters** `meta` (`ResourceDescription`) – The resource to load

**classmethod** `Loader.supports_file(meta)`

Check if the loader has a supported file extension.

What extensions are supported can be defined in the `file_extensions` class attribute.

`Loader.load()`

Loads a wavefront/obj file including materials and textures

**Returns** The Scene instance

**Return type** `Scene`

`Loader.find_data(path)`

Find resource using data finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

`Loader.find_program(path)`

Find resource using program finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

`Loader.find_texture(path)`

Find resource using texture finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

`Loader.find_scene(path)`

Find resource using scene finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

## 11.6.2 Attributes

`Loader.kind = 'wavefront'`

`Loader.file_extensions = [['.obj'], ['.obj', '.gz'], ['.bin']]`

`Loader.ctx`

ModernGL context

**Type** `moderngl.Context`

## 11.7 scene.gltf2.Loader

### 11.7.1 Method

`Loader.__init__(meta: moderngl_window.meta.scene.SceneDescription)`

Initialize loading GLTF 2 scene.

Supported formats:

- gltf json format with external resources
- gltf embedded buffers
- glb Binary format

**classmethod** `Loader.supports_file(meta)`

Check if the loader has a supported file extension.

What extensions are supported can be defined in the `file_extensions` class attribute.

`Loader.load()` → `moderngl_window.scene.scene.Scene`

Load a GLTF 2 scene including referenced textures.

**Returns** The scene instance

**Return type** `Scene`

`Loader.find_data(path)`

Find resource using data finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

`Loader.find_program(path)`

Find resource using program finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource



`Loader.find_texture(path)`

Find resource using texture finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

`Loader.find_scene(path)`

Find resource using scene finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

## 11.7.2 Loader Specific Methods

`Loader.load_gltf()`

Loads a gltf json file parsing its contents

`Loader.load_glb()`

Loads a binary gltf file parsing its contents

`Loader.load_materials()`

Load materials referenced in gltf metadata

`Loader.load_nodes()`

Load nodes referenced in gltf metadata

`Loader.load_node(meta, parent=None)`

Load a single node

`Loader.load_images()`

Load images referenced in gltf metadata

`Loader.load_textures()`

Load textures referenced in gltf metadata

`Loader.load_samplers()`

Load samplers referenced in gltf metadata

`Loader.load_meshes()`

Load meshes referenced in gltf metadata

## 11.7.3 Attributes

`Loader.kind = 'gltf'`

`Loader.file_extensions = [['.gltf'], ['.glb']]`

`Loader.ctx`

ModernGL context

**Type** `moderngl.Context`

## 11.7.4 Loader Specific Attributes

`Loader.supported_extensions = []`

Supported GLTF extensions <https://github.com/KhronosGroup/glTF/tree/master/extensions>

## 11.8 scene.stl.Loader

### 11.8.1 Method

`Loader.__init__(meta)`

Initialize loader.

Loaders take a `ResourceDescription` instance containing all the parameters needed to load and initialize this data.

**Parameters** `meta` (*ResourceDescription*) – The resource to load

**classmethod** `Loader.supports_file(meta)`

Check if the loader has a supported file extension.

What extensions are supported can be defined in the `file_extensions` class attribute.

`Loader.load()` → `moderngl_window.scene.scene.Scene`

Loads and stl scene/file

**Returns** The Scene instance

**Return type** Scene

`Loader.find_data(path)`

Find resource using data finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

`Loader.find_program(path)`

Find resource using program finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

`Loader.find_texture(path)`

Find resource using texture finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

`Loader.find_scene(path)`

Find resource using scene finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

### 11.8.2 Attributes

`Loader.kind = 'stl'`

`Loader.file_extensions = [['.stl'], ['.stl', '.gz']]`

`Loader.ctx`

ModernGL context

**Type** `moderngl.Context`

## 11.9 data.json.Loader

### 11.9.1 Method

`Loader.__init__(meta)`

Initialize loader.

Loaders take a `ResourceDescription` instance containing all the parameters needed to load and initialize this data.

**Parameters** `meta` (*ResourceDescription*) – The resource to load

**classmethod** `Loader.supports_file(meta)`

Check if the loader has a supported file extension.

What extensions are supported can be defined in the `file_extensions` class attribute.

`Loader.load()` → dict

Load a file as json

**Returns** The json contents

**Return type** dict

`Loader.find_data(path)`

Find resource using data finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

`Loader.find_program(path)`

Find resource using program finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

`Loader.find_texture(path)`

Find resource using texture finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

`Loader.find_scene(path)`

Find resource using scene finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

### 11.9.2 Attributes

`Loader.kind = 'json'`

`Loader.file_extensions = ['.json']`

`Loader.ctx`

ModernGL context

**Type** `moderngl.Context`

## 11.10 data.text.Loader

### 11.10.1 Method

`Loader.__init__(meta)`

Initialize loader.

Loaders take a `ResourceDescription` instance containing all the parameters needed to load and initialize this data.

**Parameters** `meta` (*ResourceDescription*) – The resource to load

**classmethod** `Loader.supports_file(meta)`

Check if the loader has a supported file extension.

What extensions are supported can be defined in the `file_extensions` class attribute.

`Loader.load()` → str

Load a file in text mode.

**Returns** The string contents of the file

**Return type** str

`Loader.find_data(path)`

Find resource using data finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

`Loader.find_program(path)`

Find resource using program finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

`Loader.find_texture(path)`

Find resource using texture finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

`Loader.find_scene(path)`

Find resource using scene finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

### 11.10.2 Attributes

`Loader.kind = 'text'`

`Loader.file_extensions = ['.txt']`

`Loader.ctx`

ModernGL context

**Type** `moderngl.Context`

## 11.11 data.binary.Loader

### 11.11.1 Method

`Loader.__init__(meta)`

Initialize loader.

Loaders take a `ResourceDescription` instance containing all the parameters needed to load and initialize this data.

**Parameters** `meta` (*ResourceDescription*) – The resource to load

**classmethod** `Loader.supports_file(meta)`

Check if the loader has a supported file extension.

What extensions are supported can be defined in the `file_extensions` class attribute.

`Loader.load()` → bytes

Load a file in binary mode

**Returns** The bytes contents of the file

**Return type** bytes

`Loader.find_data(path)`

Find resource using data finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

`Loader.find_program(path)`

Find resource using program finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

`Loader.find_texture(path)`

Find resource using texture finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

`Loader.find_scene(path)`

Find resource using scene finders.

This is mainly a shortcut method to simplify the task.

**Parameters** `path` – Path to resource

### 11.11.2 Attributes

`Loader.kind = 'binary'`

`Loader.file_extensions = []`

`Loader.ctx`

ModernGL context

**Type** `moderngl.Context`



## MODERNGL\_WINDOW.META

### 12.1 base.ResourceDescription

`moderngl_window.meta.base.ResourceDescription`

Description of any resource. Resource descriptions are required to load a resource. This class can be extended to add more specific properties.

#### 12.1.1 Methods

`ResourceDescription.__init__ (**kwargs)`

Initialize a resource description

**Parameters** **\*\*kwargs** – Attributes describing the resource to load

#### 12.1.2 Attributes

`ResourceDescription.path`

The path to a resource when a single file is specified

**Type** str

`ResourceDescription.resolved_path`

The resolved path by a finder.

The absolute path to the resource can optionally be assigned by a loader class.

**Type** pathlib.Path

`ResourceDescription.attrs`

All keywords arguments passed to the resource

**Type** dict

`ResourceDescription.label`

optional name for the resource

Assigning a label is not mandatory but can help when aliasing resources. Some prefer to preload all needed resources and fetch them later by the label. This can be a lot less chaotic in larger applications.

**Type** str

`ResourceDescription.kind`

default resource kind.

The resource kind is directly matched with the kind in loader classes.

This property also supports assignment and is useful if the `kind` is detected based in the the attribute values.

```
description.kind = 'something'
```

**Type** str

`ResourceDescription.loader_cls`

The loader class for this resource.

This property is assigned to during the loading stage were a loader class is assigned based on the *kind*.

**Type** Type

`ResourceDescription.default_kind = None`

The default kind for this resource type

**Type** str

`ResourceDescription.resource_type = None`

A unique identifier for the resource type

**Type** str

## 12.2 texture.TextureDescription

`moderngl_window.meta.texture.TextureDescription`

Describes a texture to load.

Example:

```
# Loading a 2d texture
TextureDescription(path='textures/wood.png')

# Loading a 2d texture with mipmapmaps with anisotropy
TextureDescription(path='textures/wood.png', mipmap=True, anisotropy=16.0)

# Loading texture array containing 10 layers
TextureDescription(path='textures/tiles.png', layers=10, kind='array')
```

### 12.2.1 Methods

`TextureDescription.__init__`(*path*: str = None, *kind*: str = None, *flip*=True, *mipmap*=False, *mipmap\_levels*: Tuple[int, int] = None, *anisotropy*=1.0, *image*=None, *layers*=None, *\*\*kwargs*)

Describes a texture resource

#### Parameters

- **path** (str) – path to resource relative to search directories
- **flip** (boolean) – Flip the image horizontally
- **mipmap** (bool) – Generate mipmaps. Will generate max possible levels unless *mipmap\_levels* is defined.
- **mipmap\_levels** (tuple) – (base, max\_level) controlling mipmap generation. When defined the *mipmap* parameter is automatically *True*.



- **anisotropy** (*float*) – Number of samples for anisotropic filtering
- **kind** (*str*) – The kind of loader to use
- **image** – PIL image for when loading embedded resources
- **layers** – (int): Number of layers for texture arrays
- **\*\*kwargs** – Any optional/custom attributes

### 12.2.2 Attributes

TextureDescription.**mipmap**

If mipmaps should be generated

**Type** bool

TextureDescription.**image**

PIL image when loading embedded resources

**Type** Image

TextureDescription.**layers**

Number of layers in texture array

**Type** int

TextureDescription.**anisotropy**

Number of samples for anisotropic filtering

**Type** float

TextureDescription.**mipmap\_levels**

base, max\_level for mipmap generation

**Type** Tuple[int, int]

TextureDescription.**flip**

If the image should be flipped horizontally

**Type** bool

### 12.2.3 Inherited Attributes

TextureDescription.**path**

The path to a resource when a single file is specified

**Type** str

TextureDescription.**resolved\_path**

The resolved path by a finder.

The absolute path to the resource can optionally be assigned by a loader class.

**Type** pathlib.Path

TextureDescription.**attrs**

All keywords arguments passed to the resource

**Type** dict

`TextureDescription.label`  
optional name for the resource

Assigning a label is not mandatory but can help when aliasing resources. Some prefer to preload all needed resources and fetch them later by the label. This can be a lot less chaotic in larger applications.

**Type** str

`TextureDescription.kind`  
default resource kind.

The resource kind is directly matched with the kind in loader classes.

This property also supports assignment and is useful if the kind is detected based on the attribute values.

```
description.kind = 'something'
```

**Type** str

`TextureDescription.loader_cls`  
The loader class for this resource.

This property is assigned to during the loading stage where a loader class is assigned based on the *kind*.

**Type** Type

`TextureDescription.default_kind = '2d'`

`TextureDescription.resource_type = 'textures'`

## 12.3 program.ProgramDescription

`moderngl_window.meta.program.ProgramDescription`  
Describes a program to load

By default a program can be loaded in the following ways:

- By supplying a *path* to a single glsl file containing all shaders
- By supplying several paths to separate files containing each shader type. For example `vertex_shader`, `fragment_shader` .. etc.

```
# Single glsl file containing all shaders
ProgramDescription(path='programs/myprogram.glsl')

# Multiple shader files
ProgramDescription(
    vertex_shader='programs/myprogram_vs.glsl',
    fragment_shader='programs/myprogram_fs.glsl',
    geometry_shader='programs/myprogram_gs.glsl',
)
```

### 12.3.1 Methods

`ProgramDescription.__init__` (*path: str = None, kind: str = None, reloadable=False, vertex\_shader: str = None, geometry\_shader: str = None, fragment\_shader: str = None, tess\_control\_shader: str = None, tess\_evaluation\_shader: str = None, \*\*kwargs*)

Create a program description

#### Keyword Arguments

- **path** (*str*) – path to the resource relative to search directories
- **kind** (*str*) – The kind of loader to use
- **reloadable** (*bool*) – Should this program be reloadable
- **vertex\_shader** (*str*) – Path to vertex shader file
- **geometry\_shader** (*str*) – Path to geometry shader
- **fragment\_shader** (*str*) – Path to fragmet shader
- **tess\_control\_shader** (*str*) –
- **tess\_evaluation\_shader** (*str*) – Path to tess eval shader
- **\*\*kwargs** – Optional custom attributes

### 12.3.2 Attributes

`ProgramDescription.tess_evaluation_shader`  
Relative path to tessellation evaluation shader

**Type** *str*

`ProgramDescription.vertex_shader`  
Relative path to vertex shader

**Type** *str*

`ProgramDescription.geometry_shader`  
Relative path to geometry shader

**Type** *str*

`ProgramDescription.reloadable`  
if this program is reloadable

**Type** *bool*

`ProgramDescription.fragment_shader`  
Relative path to fragment shader

**Type** *str*

`ProgramDescription.tess_control_shader`  
Relative path to tess control shader

**Type** *str*

### 12.3.3 Inherited Attributes

`ProgramDescription.path`

The path to a resource when a single file is specified

**Type** str

`ProgramDescription.resolved_path`

The resolved path by a finder.

The absolute path to the resource can optionally be assigned by a loader class.

**Type** pathlib.Path

`ProgramDescription.attrs`

All keywords arguments passed to the resource

**Type** dict

`ProgramDescription.label`

optional name for the resource

Assigning a label is not mandatory but can help when aliasing resources. Some prefer to preload all needed resources and fetch them later by the label. This can be a lot less chaotic in larger applications.

**Type** str

`ProgramDescription.kind`

default resource kind.

The resource kind is directly matched with the kind in loader classes.

This property also supports assignment and is useful if the kind is detected based on the attribute values.

```
description.kind = 'something'
```

**Type** str

`ProgramDescription.loader_cls`

The loader class for this resource.

This property is assigned to during the loading stage were a loader class is assigned based on the *kind*.

**Type** Type

`ProgramDescription.default_kind = None`

`ProgramDescription.resource_type = 'programs'`

## 12.4 scene.SceneDescription

`moderngl_window.meta.scene.SceneDescription`

Describes a scene to load.

The correct loader is resolved by looking at the file extension. This can be overridden by specifying a kind that maps directly to a specific loader class.

```
# Wavefront/obj file
SceneDescription(path='scenes/cube.obj')

# stl file
SceneDescription(path='scenes/crater.stl')

# GLTF 2 file
SceneDescription(path='scenes/sponza.glTF')
```

The user can also override what buffer/attribute names should be used by specifying `attr_names`.

A `cache` option is also available as some scene loaders supports converting the file into a different format on the fly to speed up loading.

### 12.4.1 Methods

`SceneDescription.__init__`(*path=None, kind=None, cache=False, attr\_names=<class 'moderngl\_window.geometry.attributes.AttributeNames'>, \*\*kwargs*)

Create a scene description.

#### Keyword Arguments

- **path** (*str*) – Path to resource
- **kind** (*str*) – Loader kind
- **cache** (*str*) – Use the loader caching system if present
- **attr\_names** (*AttributeNames*) – Attrib name config
- **\*\*kwargs** – Optional custom attributes

### 12.4.2 Attributes

`SceneDescription.attr_names`

Attribute name config

**Type** `AttributeNames`

`SceneDescription.cache`

Use cache feature in scene loader

**Type** `bool`

### 12.4.3 Inherited Attributes

`SceneDescription.path`

The path to a resource when a single file is specified

**Type** `str`

`SceneDescription.resolved_path`

The resolved path by a finder.

The absolute path to the resource can optionally be assigned by a loader class.

**Type** `pathlib.Path`

`SceneDescription.attrs`

All keywords arguments passed to the resource

**Type** dict

`SceneDescription.label`

optional name for the resource

Assigning a label is not mandatory but can help when aliasing resources. Some prefer to preload all needed resources and fetch them later by the label. This can be a lot less chaotic in larger applications.

**Type** str

`SceneDescription.kind`

default resource kind.

The resource kind is directly matched with the kind in loader classes.

This property also supports assignment and is useful if the kind is detected based in the the attribute values.

```
description.kind = 'something'
```

**Type** str

`SceneDescription.loader_cls`

The loader class for this resource.

This property is assigned to during the loading stage were a loader class is assigned based on the *kind*.

**Type** Type

`SceneDescription.default_kind = None`

`SceneDescription.resource_type = 'scenes'`

## 12.5 data.DataDescription

`moderngl_window.meta.data.DataDescription`

Describes data file to load.

This is a generic resource description type for loading resources that are not textures, programs and scenes. That loaded class is used depends on the kind or the file extension.

Currently used to load:

- text files
- json files
- binary files

```
# Describe a text file. Text loader is used based on file extension
DataDescription(path='data/text.txt')
```

```
# Describe a json file. Json loader is used based on file extension
DataDescription(path='data/data.json')
```

```
# Describe a binary file. Specify a binary loader should be used.
DataDescription(path='data/data.bin', kind='binary')
```

## 12.5.1 Methods

`DataDescription.__init__` (*path=None, kind=None, \*\*kwargs*)  
Initialize the resource description.

### Keyword Arguments

- **path** (*str*) – Relative path to the resource
- **kind** (*str*) – The resource kind deciding loader class
- **\*\*kwargs** – Additional custom attributes

## 12.5.2 Attributes

`DataDescription.path`  
The path to a resource when a single file is specified

**Type** `str`

`DataDescription.resolved_path`  
The resolved path by a finder.

The absolute path to the resource can optionally be assigned by a loader class.

**Type** `pathlib.Path`

`DataDescription.attrs`  
All keywords arguments passed to the resource

**Type** `dict`

`DataDescription.label`  
optional name for the resource

Assigning a label is not mandatory but can help when aliasing resources. Some prefer to preload all needed resources and fetch them later by the label. This can be a lot less chaotic in larger applications.

**Type** `str`

`DataDescription.kind`  
default resource kind.

The resource kind is directly matched with the `kind` in loader classes.

This property also supports assignment and is useful if the `kind` is detected based in the the attribute values.

```
description.kind = 'something'
```

**Type** `str`

`DataDescription.loader_cls`  
The loader class for this resource.

This property is assigned to during the loading stage were a loader class is assigned based on the *kind*.

**Type** `Type`

`DataDescription.default_kind = None`

`DataDescription.resource_type = 'data'`





## MODERNGL\_WINDOW.FINDERS

### 13.1 base.BaseFileSystemFinder

`moderngl_window.finders.base.BaseFileSystemFinder`  
Base class for searching filesystem directories

#### 13.1.1 Methods

`BaseFileSystemFinder.__init__()`  
Initialize finder class by looking up the paths referenced in `settings_attr`.

`BaseFileSystemFinder.find(path: pathlib.Path) → pathlib.Path`  
Finds a file in the configured paths returning its absolute path.

**Parameters** `path` (*pathlib.Path*) – The path to find

**Returns** The absolute path to the file or None if not found

#### 13.1.2 Attributes

`BaseFileSystemFinder.settings_attr = None`  
Name of the attribute in *Settings* containing a list of paths the finder should search in.  
**Type** str

### 13.2 texture.FileSystemFinder

`moderngl_window.finders.texture.FileSystemFinder`  
Find textures in `settings.TEXTURE_DIRS`

#### 13.2.1 Methods

`FileSystemFinder.__init__()`  
Initialize finder class by looking up the paths referenced in `settings_attr`.

`FileSystemFinder.find(path: pathlib.Path) → pathlib.Path`  
Finds a file in the configured paths returning its absolute path.

**Parameters** `path` (*pathlib.Path*) – The path to find

**Returns** The absolute path to the file or None if not found

### 13.2.2 Attributes

`FileSystemFinder.settings_attr = 'TEXTURE_DIRS'`

## 13.3 program.FileSystemFinder

`moderngl_window.finders.program.FileSystemFinder`  
Find shaders in `settings.PROGRAM_DIRS`

### 13.3.1 Methods

`FileSystemFinder.__init__()`  
Initialize finder class by looking up the paths referenced in `settings_attr`.

`FileSystemFinder.find(path: pathlib.Path) → pathlib.Path`  
Finds a file in the configured paths returning its absolute path.

**Parameters** `path` (*pathlib.Path*) – The path to find

**Returns** The absolute path to the file or None if not found

### 13.3.2 Attributes

`FileSystemFinder.settings_attr = 'PROGRAM_DIRS'`

## 13.4 scene.FileSystemFinder

`moderngl_window.finders.scene.FileSystemFinder`  
Find scenes in `settings.SCENE_DIRS`

### 13.4.1 Methods

`FileSystemFinder.__init__()`  
Initialize finder class by looking up the paths referenced in `settings_attr`.

`FileSystemFinder.find(path: pathlib.Path) → pathlib.Path`  
Finds a file in the configured paths returning its absolute path.

**Parameters** `path` (*pathlib.Path*) – The path to find

**Returns** The absolute path to the file or None if not found

### 13.4.2 Attributes

`FileSystemFinder.settings_attr = 'SCENE_DIRS'`

## 13.5 data.FilesystemFinder

moderngl\_window.finders.data.**FilesystemFinder**

Find data in settings.DATA\_DIRS

### 13.5.1 Methods

`FilesystemFinder.__init__()`

Initialize finder class by looking up the paths referenced in `settings_attr`.

`FilesystemFinder.find(path: pathlib.Path) → pathlib.Path`

Finds a file in the configured paths returning its absolute path.

**Parameters** `path` (*pathlib.Path*) – The path to find

**Returns** The absolute path to the file or None if not found

### 13.5.2 Attributes

`FilesystemFinder.settings_attr = 'DATA_DIRS'`



## MODERNGL\_WINDOW.OPENGL

### 14.1 opengl.projection.Projection3D

`moderngl_window.opengl.projection.Projection3D`  
3D Projection

#### 14.1.1 Methods

`Projection3D.__init__` (*aspect\_ratio=1.7777777777777777, fov=75.0, near=1.0, far=100.0*)  
Create a 3D projection

##### Keyword Arguments

- **aspect\_ratio** (*float*) – Aspect ratio
- **fov** (*float*) – Field of view
- **near** (*float*) – Near plane value
- **far** (*float*) – Far plane value

`Projection3D.update` (*aspect\_ratio: float = None, fov: float = None, near: float = None, far: float = None*) → *None*  
Update the projection matrix

##### Keyword Arguments

- **aspect\_ratio** (*float*) – Aspect ratio
- **fov** (*float*) – Field of view
- **near** (*float*) – Near plane value
- **far** (*float*) – Far plane value

`Projection3D.tobytes` () → *bytes*  
Get the byte representation of the projection matrix

**Returns** byte representation of the projection matrix

**Return type** *bytes*

#### 14.1.2 Attributes

`Projection3D.aspect_ratio`  
The projection's aspect ratio

**Type** float

`Projection3D.fov`

Current field of view

**Type** float

`Projection3D.near`

Current near plane value

**Type** float

`Projection3D.far`

Current far plane value

**Type** float

`Projection3D.matrix`

Current numpy projection matrix

**Type** np.ndarray

`Projection3D.projection_constants`

(x, y) projection constants for the current projection. This is for example useful when reconstructing a view position of a fragment from a linearized depth value.

## 14.2 opengl.vao.VAO

`moderngl_window.opengl.vao.VAO`

Represents a vertex array object.

This is a wrapper class over `moderngl.VertexArray` to make interactions with programs/shaders simpler. Named buffers are added corresponding with attribute names in a vertex shader. When rendering the VAO an internal `moderngl.VertexArray` is created by automatically creating a buffer mapping compatible with the supplied program. This program is cached internally.

The shader program doesn't need to use all the buffers registered in this wrapper. When a subset is used only the used buffers are mapped and the appropriate padding is calculated when interleaved data is used.

There is no requirements to use this class, but most methods in the system creating vertexbuffers will return this type. You can obtain a single `moderngl.VertexBuffer` instance by calling `VAO.instance()` method if you prefer to work directly on moderngl instances.

Example:

```
# Separate buffers
vao = VAO(name="test", mode=moderngl.POINTS)
vao.buffer(positions, '3f', ['in_position'])
vao.buffer(velocities, '3f', ['in_velocities'])

# Interleaved
vao = VAO(name="test", mode=moderngl.POINTS)
vao.buffer(interleaved_data, '3f 3f', ['in_position', 'in_velocities'])
```

```
# GLSL vertex shader in attributes
in vec3 in_position;
in vec3 in_velocities;
```

### 14.2.1 Methods

`VAO.__init__(name="", mode=4)`

Create and empty VAO with a name and default render mode.

Example:

```
VAO(name="cube", mode=moderngl.TRIANGLES)
```

#### Keyword Arguments

- **name** (*str*) – Optional name for debug purposes
- **mode** (*int*) – Default draw mode

`VAO.render(program: moderngl.program.Program, mode=None, vertices=-1, first=0, instances=1)`

Render the VAO.

An internal `moderngl.VertexBuffer` with compatible buffer bindings is automatically created on the fly and cached internally.

**Parameters** `program` – The `moderngl.Program`

#### Keyword Arguments

- **mode** – Override the draw mode (TRIANGLES etc)
- **vertices** (*int*) – The number of vertices to transform
- **first** (*int*) – The index of the first vertex to start with
- **instances** (*int*) – The number of instances

`VAO.render_indirect(program: moderngl.program.Program, buffer, mode=None, count=-1, *, first=0)`

The render primitive (mode) must be the same as the input primitive of the GeometryShader. The draw commands are 5 integers: (count, instanceCount, firstIndex, baseVertex, baseInstance).

#### Parameters

- **program** – The `moderngl.Program`
- **buffer** – The `moderngl.Buffer` containing indirect draw commands

#### Keyword Arguments

- **mode** (*int*) – By default TRIANGLES will be used.
- **count** (*int*) – The number of draws.
- **first** (*int*) – The index of the first indirect draw command.

`VAO.transform(program: moderngl.program.Program, buffer: moderngl.buffer.Buffer, mode=None, vertices=-1, first=0, instances=1)`

Transform vertices. Stores the output in a single buffer.

#### Parameters

- **program** – The `moderngl.Program`
- **buffer** – The `moderngl.buffer` to store the output

#### Keyword Arguments

- **mode** – Draw mode (for example `moderngl.POINTS`)
- **vertices** (*int*) – The number of vertices to transform

- **first** (*int*) – The index of the first vertex to start with
- **instances** (*int*) – The number of instances

VAO.**buffer** (*buffer, buffer\_format: str, attribute\_names: List[str]*)

Register a buffer/vbo for the VAO. This can be called multiple times. adding multiple buffers (interleaved or not).

**Parameters**

- **buffer** – The buffer data. Can be `numpy.array`, `moderngl.Buffer` or `bytes`.
- **buffer\_format** (*str*) – The format of the buffer. (eg. `3f 3f` for interleaved positions and normals).
- **attribute\_names** – A list of attribute names this buffer should map to.

**Returns** The `moderngl.Buffer` instance object. This is handy when providing `bytes` and `numpy.array`.

VAO.**index\_buffer** (*buffer, index\_element\_size=4*)

Set the index buffer for this VAO.

**Parameters** **buffer** – `moderngl.Buffer`, `numpy.array` or `bytes`

**Keyword Arguments** **index\_element\_size** (*int*) – Byte size of each element. 1, 2 or 4

VAO.**instance** (*program: moderngl.program.Program*) → `moderngl.vertex_array.VertexArray`

Obtain the `moderngl.VertexArray` instance for the program.

The instance is only created once and cached internally.

**Parameters** **program** (*moderngl.Program*) – The program

**Returns** instance

**Return type** `moderngl.VertexArray`

VAO.**release** (*buffer=True*)

Destroy all internally cached vaos and release all buffers.

**Keyword Arguments** **buffers** (*bool*) – also release buffers

VAO.**get\_buffer\_by\_name** (*name: str*) → `moderngl_window.opengl.vao.BufferInfo`

Get the `BufferInfo` associated with a specific attribute name

If no buffer is associated with the name *None* will be returned.

**Parameters** **name** (*str*) – Name of the mapped attribute

**Returns** `BufferInfo` instance

**Return type** `BufferInfo`

## 14.2.2 Attributes

VAO.**ctx**

The active `moderngl` context

**Type** `moderngl.Context`



## MODERNGL\_WINDOW.RESOURCES

`moderngl_window.resources.register_dir` (*path: Union[pathlib.Path, str]*) → None

Adds a resource directory for all resource types

**Parameters** *path* (*Union[Path, str]*) – Directory path

`moderngl_window.resources.register_program_dir` (*path: Union[pathlib.Path, str]*) → None

Adds a resource directory specifically for programs

**Parameters** *path* (*Union[Path, str]*) – Directory path

`moderngl_window.resources.register_texture_dir` (*path: Union[pathlib.Path, str]*) → None

Adds a resource directory specifically for textures

**Parameters** *path* (*Union[Path, str]*) – Directory path

`moderngl_window.resources.register_scene_dir` (*path: Union[pathlib.Path, str]*) → None

Adds a resource directory specifically for scenes

**Parameters** *path* (*Union[Path, str]*) – Directory path

`moderngl_window.resources.register_data_dir` (*path: Union[pathlib.Path, str]*) → None

Adds a resource directory specifically for data files

**Parameters** *path* (*Union[Path, str]*) – Directory path

### 15.1 base.BaseRegistry

`moderngl_window.resources.base.BaseRegistry`

Base class for all resource pools

#### 15.1.1 Methods

`BaseRegistry.__init__()`

Initialize internal attributes

`BaseRegistry.load` (*meta: moderngl\_window.meta.base.ResourceDescription*) → Any

Loads a resource using the configured finders and loaders.

**Parameters** *meta* (*ResourceDescription*) – The resource description

`BaseRegistry.add` (*meta: moderngl\_window.meta.base.ResourceDescription*) → None

Adds a resource description without loading it. The resource is loaded and returned when `load_pool()` is called.

**Parameters** *meta* (*ResourceDescription*) – The resource description

`BaseRegistry.load_pool()` → `Generator[Tuple[moderngl_window.meta.base.ResourceDescription, Any], None, None]`

Loads all the data files using the configured finders.

This is only relevant when resource have been added to this pool using `add()`.

**Returns** Generator of (meta, resource) tuples

`BaseRegistry.resolve_loader(meta: moderngl_window.meta.base.ResourceDescription)` → `None`

Attempts to assign a loader class to a `ResourceDescription`.

**Parameters** `meta` (*ResourceDescription*) – The resource description instance

## 15.1.2 Attributes

`BaseRegistry.settings_attr = None`

The name of the attribute in *Settings* containing a list of loader classes.

**Type** str

`BaseRegistry.count`

The number of resource descriptions added. This is only relevant when using *add* and *load\_pool*.

**Type** int

`BaseRegistry.loaders`

Loader classes for this resource type

**Type** Generator

## 15.2 textures.Textures

`moderngl_window.resources.textures.Textures`

Handles texture resources

### 15.2.1 Methods

`Textures.__init__()`

Initialize internal attributes

`Textures.load(meta: moderngl_window.meta.texture.TextureDescription)` →

`Union[moderngl.texture.Texture, moderngl.texture_array.TextureArray]`

Loads a texture with the configured loaders.

**Parameters** `meta` (*TextureDescription*) – The resource description

**Returns** 2d texture

**Return type** `moderngl.Texture`

**Returns** texture array if *layers* is supplied

**Return type** `moderngl.TextureArray`

`Textures.add(meta: moderngl_window.meta.base.ResourceDescription)` → `None`

Adds a resource description without loading it. The resource is loaded and returned when `load_pool()` is called.

**Parameters** `meta` (*ResourceDescription*) – The resource description

`Textures.load_pool()` → Generator[Tuple[moderngl\_window.meta.base.ResourceDescription, Any], None, None]

Loads all the data files using the configured finders.

This is only relevant when resource have been added to this pool using `add()`.

**Returns** Generator of (meta, resource) tuples

`Textures.resolve_loader(meta: moderngl_window.meta.base.ResourceDescription)` → None

Attempts to assign a loader class to a ResourceDescription.

**Parameters** `meta` (*ResourceDescription*) – The resource description instance

## 15.2.2 Attributes

`Textures.settings_attr = 'TEXTURE_LOADERS'`

`Textures.count`

The number of resource descriptions added. This is only relevant when using *add* and *load\_pool*.

**Type** int

`Textures.loaders`

Loader classes for this resource type

**Type** Generator

## 15.3 programs.Programs

`moderngl_window.resources.programs.Programs`

Handle program loading

### 15.3.1 Methods

`Programs.__init__()`

Initialize internal attributes

`Programs.load(meta: moderngl_window.meta.program.ProgramDescription)` → moderngl.program.Program

Loads a shader program with the configured loaders

**Parameters** `meta` (*ProgramDescription*) – The resource description

**Returns** The shader program

**Return type** moderngl.Program

`Programs.add(meta: moderngl_window.meta.base.ResourceDescription)` → None

Adds a resource description without loading it. The resource is loaded and returned when `load_pool()` is called.

**Parameters** `meta` (*ResourceDescription*) – The resource description

`Programs.load_pool()` → Generator[Tuple[moderngl\_window.meta.base.ResourceDescription, Any], None, None]

Loads all the data files using the configured finders.

This is only relevant when resource have been added to this pool using `add()`.

**Returns** Generator of (meta, resource) tuples

`Programs.resolve_loader` (*meta: moderngl\_window.meta.program.ProgramDescription*) → None  
 Resolve program loader.

Determines if the references resource is a single or multiple glsl files unless `kind` is specified.

**Parameters** *meta* (*ProgramDescription*) – The resource description

### 15.3.2 Attributes

`Programs.settings_attr` = 'PROGRAM\_LOADERS'

`Programs.count`

The number of resource descriptions added. This is only relevant when using `add` and `load_pool`.

**Type** int

`Programs.loaders`

Loader classes for this resource type

**Type** Generator

## 15.4 scenes.Scenes

`moderngl_window.resources.scenes.Scenes`

Handles scene loading

### 15.4.1 Methods

`Scenes.__init__()`

Initialize internal attributes

`Scenes.load` (*meta: moderngl\_window.meta.scene.SceneDescription*) → `moderngl_window.scene.scene.Scene`

Load a scene with the configured loaders.

**Parameters** *meta* (*SceneDescription*) – The resource description

**Returns** The loaded scene

**Return type** Scene

`Scenes.add` (*meta: moderngl\_window.meta.base.ResourceDescription*) → None

Adds a resource description without loading it. The resource is loaded and returned when `load_pool()` is called.

**Parameters** *meta* (*ResourceDescription*) – The resource description

`Scenes.load_pool()` → Generator[Tuple[moderngl\_window.meta.base.ResourceDescription, Any], None, None]

Loads all the data files using the configured finders.

This is only relevant when resource have been added to this pool using `add()`.

**Returns** Generator of (meta, resource) tuples

`Scenes.resolve_loader` (*meta: moderngl\_window.meta.base.ResourceDescription*) → None

Attempts to assign a loader class to a ResourceDescription.

**Parameters** *meta* (*ResourceDescription*) – The resource description instance

## 15.4.2 Attributes

`Scenes.settings_attr = 'SCENE_LOADERS'`

`Scenes.count`

The number of resource descriptions added. This is only relevant when using *add* and *load\_pool*.

**Type** int

`Scenes.loaders`

Loader classes for this resource type

**Type** Generator

## 15.5 base.DataFiles

`moderngl_window.resources.data.DataFiles`

Registry for requested data files

### 15.5.1 Methods

`DataFiles.__init__()`

Initialize internal attributes

`DataFiles.load(meta: moderngl_window.meta.data.DataDescription) → Any`

Load data file with the configured loaders.

**Parameters** `meta` (*DataDescription*) – the resource description

**Returns** The loaded resource

**Return type** Any

`DataFiles.add(meta: moderngl_window.meta.base.ResourceDescription) → None`

Adds a resource description without loading it. The resource is loaded and returned when `load_pool()` is called.

**Parameters** `meta` (*ResourceDescription*) – The resource description

`DataFiles.load_pool() → Generator[Tuple[moderngl_window.meta.base.ResourceDescription, Any], None, None]`

Loads all the data files using the configured finders.

This is only relevant when resource have been added to this pool using `add()`.

**Returns** Generator of (meta, resource) tuples

`DataFiles.resolve_loader(meta: moderngl_window.meta.base.ResourceDescription) → None`

Attempts to assign a loader class to a ResourceDescription.

**Parameters** `meta` (*ResourceDescription*) – The resource description instance

### 15.5.2 Attributes

`DataFiles.settings_attr = 'DATA_LOADERS'`

`DataFiles.count`

The number of resource descriptions added. This is only relevant when using *add* and *load\_pool*.

**Type** int

DataFiles.**loaders**

Loader classes for this resource type

**Type** Generator

## MODERNGL\_WINDOW.TIMERS

### 16.1 base.BaseTimer

`moderngl_window.timers.base.BaseTimer`

A timer controls the time passed into the the render function. This can be used in creative ways to control the current time such as basing it on current location in an audio file.

All methods must be implemented.

#### 16.1.1 Methods

`BaseTimer.__init__()`

Initialize self. See `help(type(self))` for accurate signature.

`BaseTimer.next_frame()` → `Tuple[float, float]`

Get timer information for the next frame.

**Returns** The frametime and current time

**Return type** `Tuple[float, float]`

`BaseTimer.start()`

Start the timer initially or resume after pause

`BaseTimer.pause()`

Pause the timer

`BaseTimer.toggle_pause()`

Toggle pause state

`BaseTimer.stop()` → `Tuple[float, float]`

Stop the timer. Should only be called once when stopping the timer.

**Returns** `Tuple[float, float]`> Current position in the timer, actual running duration

#### 16.1.2 Attributes

`BaseTimer.is_paused`

The pause state of the timer

**Type** `bool`

`BaseTimer.is_running`

Is the timer currently running?

**Type** bool

`BaseTimer.time`

Get or set the current time. This can be used to jump around in the timeline.

**Returns** The current time in seconds

**Return type** float

## 16.2 clock.Timer

`moderngl_window.timers.clock.Timer`

Timer based on python `time`.

### 16.2.1 Methods

`Timer.__init__ (**kwargs)`

Initialize self. See `help(type(self))` for accurate signature.

`Timer.next_frame ()` → `Tuple[float, float]`

Get the time and frametime for the next frame. This should only be called once per frame.

**Returns** current time and frametime

**Return type** `Tuple[float, float]`

`Timer.start ()`

Start the timer by recoding the current `time.time ()` preparing to report the number of seconds since this timestamp.

`Timer.pause ()`

Pause the timer by setting the internal pause time using `time.time ()`

`Timer.toggle_pause ()`

Toggle the paused state

`Timer.stop ()` → `Tuple[float, float]`

Stop the timer. Should only be called once when stopping the timer.

**Returns** Current position in the timer, actual running duration

**Return type** `Tuple[float, float]`

### 16.2.2 Attributes

`Timer.is_paused`

The pause state of the timer

**Type** bool

`Timer.is_running`

Is the timer currently running?

**Type** bool

`Timer.time`

Get or set the current time. This can be used to jump around in the timeline.

**Returns** The current time in seconds



## MODERNGL\_WINDOW.SCENE

### 17.1 Camera

`moderngl_window.scene.Camera`

Simple camera class containing projection.

```
# create a camera
camera = Camera(fov=60.0, aspect_ratio=1.0, near=1.0, far=100.0)

# Get the current camera matrix as numpy array
print(camera.matrix)

# Get projection matrix as numpy array
print(camera.projection.matrix)
```

#### 17.1.1 Methods

`Camera.__init__(fov=60.0, aspect_ratio=1.0, near=1.0, far=100.0)`

Initialize camera using a specific projection

##### Keyword Arguments

- **fov** (*float*) – Field of view
- **aspect\_ratio** (*float*) – Aspect ratio
- **near** (*float*) – Near plane
- **far** (*float*) – Far plane

`Camera.set_position(x, y, z) → None`

Set the 3D position of the camera.

##### Parameters

- **x** (*float*) – x position
- **y** (*float*) – y position
- **z** (*float*) – z position

`Camera.look_at(vec=None, pos=None) → numpy.ndarray`

Look at a specific point

Either `vec` or `pos` needs to be supplied.

##### Keyword Arguments

- **vec** (*pyrr.Vector3*) – position
- **pos** (*tuple/list*) – list of tuple `[x, y, z] / (x, y, z)`

**Returns** Camera matrix

**Return type** `numpy.ndarray`

### 17.1.2 Attributes

`Camera.matrix`

The current view matrix for the camera

**Type** `numpy.ndarray`

`Camera.projection`

The 3D projection

**Type** *Projection3D*

## 17.2 KeyboardCamera

`moderngl_window.scene.KeyboardCamera`

Camera controlled by mouse and keyboard. The class interacts with the key constants in the built in window types.

Creating a keyboard camera:

```
camera = KeyboardCamera(  
    self.wnd.keys,  
    fov=75.0,  
    aspect_ratio=self.wnd.aspect_ratio,  
    near=0.1,  
    far=1000.0,  
)
```

We can also interact with the belonging *Projection3D* instance.

```
# Update aspect ratio  
camera.projection.update(aspect_ratio=1.0)  
  
# Get projection matrix in bytes (f4)  
camera.projection.tobytes()
```

### 17.2.1 Methods

`KeyboardCamera.__init__` (*keys*: `moderngl_window.context.base.keys.BaseKeys`, *fov*=60.0, *aspect\_ratio*=1.0, *near*=1.0, *far*=100.0)

Initialize the camera

**Parameters** **keys** (*BaseKeys*) – The key constants for the current window type

**Keyword Arguments**

- **fov** (*float*) – Field of view
- **aspect\_ratio** (*float*) – Aspect ratio

- **near** (*float*) – near plane
- **far** (*float*) – far plane

KeyboardCamera.**key\_input** (*key, action, modifiers*) → None

Process key inputs and move camera

**Parameters**

- **key** – The key
- **action** – key action release/press
- **modifiers** – key modifier states such as ctrl or shift

KeyboardCamera.**set\_position** (*x, y, z*) → None

Set the 3D position of the camera.

**Parameters**

- **x** (*float*) – x position
- **y** (*float*) – y position
- **z** (*float*) – z position

KeyboardCamera.**look\_at** (*vec=None, pos=None*) → numpy.ndarray

Look at a specific point

Either *vec* or *pos* needs to be supplied.

**Keyword Arguments**

- **vec** (*pyrr.Vector3*) – position
- **pos** (*tuple/list*) – list of tuple [*x, y, z*] / (*x, y, z*)

**Returns** Camera matrix

**Return type** numpy.ndarray

KeyboardCamera.**move\_left** (*activate*) → None

The camera should be continuously moving to the left.

**Parameters** **activate** (*bool*) – Activate or deactivate this state

KeyboardCamera.**move\_right** (*activate*) → None

The camera should be continuously moving to the right.

**Parameters** **activate** (*bool*) – Activate or deactivate this state

KeyboardCamera.**move\_forward** (*activate*) → None

The camera should be continuously moving forward.

**Parameters** **activate** (*bool*) – Activate or deactivate this state

KeyboardCamera.**move\_backward** (*activate*) → None

The camera should be continuously moving backwards.

**Parameters** **activate** (*bool*) – Activate or deactivate this state

KeyboardCamera.**move\_up** (*activate*) → None

The camera should be continuously moving up.

**Parameters** **activate** (*bool*) – Activate or deactivate this state

KeyboardCamera.**move\_down** (*activate*)

The camera should be continuously moving down.

**Parameters** **activate** (*bool*) – Activate or deactivate this state

`KeyboardCamera.move_state` (*direction, activate*) → None

Set the camera position move state.

**Parameters**

- **direction** – What direction to update
- **activate** – Start or stop moving in the direction

`KeyboardCamera.rot_state` (*dx: int, dy: int*) → None

Update the rotation of the camera.

This is done by passing in the relative mouse movement change on x and y (delta x, delta y).

In the past this method took the viewport position of the mouse. This does not work well when mouse exclusivity mode is enabled.

**Parameters**

- **dx** – Relative mouse position change on x
- **dy** – Relative mouse position change on y

## 17.2.2 Attributes

`KeyboardCamera.matrix`

The current view matrix for the camera

**Type** `numpy.ndarray`

`KeyboardCamera.mouse_sensitivity`

Mouse sensitivity (rotation speed).

This property can also be set:

```
camera.mouse_sensitivity = 2.5
```

**Type** `float`

`KeyboardCamera.velocity`

The speed this camera move based on key inputs

The property can also be modified:

```
camera.velocity = 5.0
```

**Type** `float`

`KeyboardCamera.projection`

The 3D projection

**Type** `Projection3D`

## 17.3 Scene

### 17.3.1 Methods

`Scene.__init__ (name, **kwargs)`

Create a scene with a name.

**Parameters** `name` (*str*) – Unique name or path for the scene

`Scene.draw (projection_matrix: numpy.ndarray = None, camera_matrix: numpy.ndarray = None, time=0.0) → None`

Draw all the nodes in the scene.

**Parameters**

- **projection\_matrix** (*ndarray*) – projection matrix (bytes)
- **camera\_matrix** (*ndarray*) – camera\_matrix (bytes)
- **time** (*float*) – The current time

`Scene.draw_bbox (projection_matrix=None, camera_matrix=None, children=True) → None`

Draw scene and mesh bounding boxes.

**Parameters**

- **projection\_matrix** (*ndarray*) – mat4 projection
- **camera\_matrix** (*ndarray*) – mat4 camera matrix
- **children** (*bool*) – Will draw bounding boxes for meshes as well

`Scene.apply_mesh_programs (mesh_programs=None) → None`

Applies mesh programs to meshes. If not mesh programs are passed in we assign default ones.

**Parameters** `mesh_programs` (*list*) – List of mesh programs to assign

`Scene.calc_scene_bbox () → None`

Calculate scene bbox

`Scene.prepare () → None`

prepare the scene for rendering.

Calls `apply_mesh_programs ()` assigning default meshprograms if needed and sets the model matrix.

`Scene.destroy () → None`

Destroys the scene data and vertex buffers

### 17.3.2 Attributes

`Scene.ctx`

The current context

**Type** `moderngl.Context`

`Scene.model_matrix`

The current model matrix

This property is settable.

**Type** `numpy.ndarray`

## 17.4 Node

`moderngl_window.scene.Node`

A generic scene node containing a mesh or camera and/or a container for other nodes. Nodes and their children represents the scene tree.

### 17.4.1 Methods

`Node.__init__ (camera=None, mesh=None, matrix=None)`  
Create a node.

#### Keyword Arguments

- **camera** – Camera to store in the node
- **mesh** – Mesh to store in the node
- **matrix** – The node's matrix

`Node.add_child (node)`  
Add a child to this node

**Parameters** `node` (`Node`) – Node to add as a child

`Node.draw (projection_matrix=None, camera_matrix=None, time=0)`  
Draw node and children.

#### Keyword Arguments

- **projection\_matrix** (`bytes`) – projection matrix
- **camera\_matrix** (`bytes`) – camera\_matrix
- **time** (`float`) – The current time

`Node.draw_bbox (projection_matrix, camera_matrix, program, vao)`  
Draw bounding box around the node and children.

#### Keyword Arguments

- **projection\_matrix** (`bytes`) – projection matrix
- **camera\_matrix** (`bytes`) – camera\_matrix
- **program** (`moderngl.Program`) – The program to render the bbox
- **vao** – The vertex array representing the bounding box

`Node.calc_global_bbox (view_matrix, bbox_min, bbox_max)`  
Recursive calculation of scene bbox.

#### Keyword Arguments

- **view\_matrix** (`numpy.ndarray`) – view matrix
- **bbox\_min** – min bbox values
- **bbox\_max** – max bbox values

`Node.calc_model_mat (model_matrix)`  
Calculate the model matrix related to all parents.

**Parameters** `model_matrix` (`numpy.ndarray`) – model matrix

## 17.4.2 Attributes

`Node.children`  
List of children  
**Type** list

## 17.5 Mesh

`moderngl_window.scene.Mesh` = <class 'moderngl\_window.scene.mesh.Mesh'>  
Mesh info and geometry

### 17.5.1 Methods

`Mesh.__init__` (*name*, *vao=None*, *material=None*, *attributes=None*, *bbox\_min=None*, *bbox\_max=None*)  
Initialize mesh.

**Parameters** *name* (*str*) – name of the mesh

**Keyword Arguments**

- **vao** (*VAO*) – geometry
- **material** (*Material*) – material for the mesh
- **attributes** (*dict*) – Details info about each mesh attribute (dict)
- **bbox\_min** – xyz min values
- **bbox\_max** – xyz max values

Attributes example:

```
{
  "NORMAL": {"name": "in_normal", "components": 3, "type": GL_FLOAT},
  "POSITION": {"name": "in_position", "components": 3, "type": GL_FLOAT}
}
```

`Mesh.draw` (*projection\_matrix=None*, *model\_matrix=None*, *camera\_matrix=None*, *time=0.0*)  
Draw the mesh using the assigned mesh program

**Keyword Arguments**

- **projection\_matrix** (*bytes*) – projection\_matrix
- **view\_matrix** (*bytes*) – view\_matrix
- **camera\_matrix** (*bytes*) – camera\_matrix

`Mesh.draw_bbox` (*proj\_matrix*, *model\_matrix*, *cam\_matrix*, *program*, *vao*)  
Renders the bounding box for this mesh.

**Parameters**

- **proj\_matrix** – Projection matrix
- **model\_matrix** – View/model matrix
- **cam\_matrix** – Camera matrix
- **program** – The moderngl.Program rendering the bounding box

- **vao** – The vao mesh for the bounding box

`Mesh.add_attribute(attr_type, name, components)`

Add metadata about the mesh :param attr\_type: POSITION, NORMAL etc :param name: The attribute name used in the program :param components: Number of floats

`Mesh.calc_global_bbox(view_matrix, bbox_min, bbox_max)`

Calculates the global bounding.

**Parameters**

- **view\_matrix** – View matrix
- **bbox\_min** – xyz min
- **bbox\_max** – xyz max

**Returns** Combined bbox

**Return type** bbox\_min, bbox\_max

`Mesh.has_normals()` → bool

**Returns** Does the mesh have a normals?

**Return type** bool

`Mesh.has_uvs(layer=0)` → bool

**Returns** Does the mesh have texture coordinates?

**Return type** bool

## 17.6 Material

`moderngl_window.scene.Material`

Generic material

### 17.6.1 Methods

`Material.__init__(name)`

Initialize material.

**Parameters** **name** (*str*) – Name of the material

### 17.6.2 Attributes

`Material.name`

Name of the material

**Type** str

`Material.color`

RGBA color

**Type** Tuple[float, float, float, float]

`Material.mat_texture`

instance

**Type** MaterialTexture



`Material.double_sided`  
 Material surface is double sided?  
**Type** bool

## 17.7 MaterialTexture

`moderngl_window.scene.MaterialTexture`  
 Wrapper for textures used in materials. Contains a texture and a sampler object.

### 17.7.1 Methods

`MaterialTexture.__init__`(*texture: moderngl.texture.Texture = None, sampler: moderngl.sampler.Sampler = None*)  
 Initialize instance.

#### Parameters

- **texture** (*moderngl.Texture*) – Texture instance
- **sampler** (*moderngl.Sampler*) – Sampler instance

### 17.7.2 Attributes

`MaterialTexture.texture`  
 Texture instance  
**Type** `moderngl.Texture`

`MaterialTexture.sampler`  
 Sampler instance  
**Type** `moderngl.Sampler`

## 17.8 MeshProgram

`moderngl_window.scene.MeshProgram`  
 Describes how a mesh is rendered using a specific shader program

### 17.8.1 Methods

`MeshProgram.__init__`(*program: moderngl.program.Program = None, \*\*kwargs*)  
 Initialize.

**Parameters** **program** – The moderngl program

`MeshProgram.draw`(*mesh, projection\_matrix: numpy.ndarray = None, model\_matrix: numpy.ndarray = None, camera\_matrix: numpy.ndarray = None, time=0.0*)  
 Draw code for the mesh

**Parameters** **mesh** (*Mesh*) – The mesh to render

#### Keyword Arguments

- **projection\_matrix** (*numpy.ndarray*) – projection\_matrix (bytes)
- **model\_matrix** (*numpy.ndarray*) – view\_matrix (bytes)
- **camera\_matrix** (*numpy.ndarray*) – camera\_matrix (bytes)
- **time** (*float*) – The current time

`MeshProgram.apply` (*mesh*)

Determine if this `MeshProgram` should be applied to the mesh. Can return self or some `MeshProgram` instance to support dynamic `MeshProgram` creation

**Parameters** `mesh` – The mesh to inspect

## 17.8.2 Attributes

`MeshProgram.ctx`

The current context

**Type** `moderngl.Context`

## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### m

- `moderngl_window`, 17
- `moderngl_window.conf`, 20
- `moderngl_window.context.base.window`, 27
- `moderngl_window.context.glfw.window`, 38
- `moderngl_window.context.headless.window`, 45
- `moderngl_window.context.pyglet.window`, 51
- `moderngl_window.context.pyqt5.window`, 58
- `moderngl_window.context.pyside2.window`, 64
- `moderngl_window.context.sdl2.window`, 71
- `moderngl_window.finders.base`, 105
- `moderngl_window.finders.data`, 106
- `moderngl_window.finders.program`, 106
- `moderngl_window.finders.scene`, 106
- `moderngl_window.finders.texture`, 105
- `moderngl_window.geometry`, 77
- `moderngl_window.loaders.base`, 81
- `moderngl_window.loaders.data.binary`, 92
- `moderngl_window.loaders.data.json`, 90
- `moderngl_window.loaders.data.text`, 91
- `moderngl_window.loaders.program.separate`, 85
- `moderngl_window.loaders.program.single`, 83
- `moderngl_window.loaders.scene.glTF2`, 88
- `moderngl_window.loaders.scene.stl`, 89
- `moderngl_window.loaders.scene.wavefront`, 87
- `moderngl_window.loaders.texture.array`, 86
- `moderngl_window.loaders.texture.t2d`, 82
- `moderngl_window.meta.base`, 95
- `moderngl_window.meta.data`, 102
- `moderngl_window.meta.program`, 98
- `moderngl_window.meta.scene`, 100
- `moderngl_window.meta.texture`, 96
- `moderngl_window.opengl.projection`, 109
- `moderngl_window.opengl.vao`, 110
- `moderngl_window.resources`, 112
- `moderngl_window.resources.base`, 113
- `moderngl_window.resources.data`, 117
- `moderngl_window.resources.programs`, 115
- `moderngl_window.resources.scenes`, 116
- `moderngl_window.resources.textures`, 114
- `moderngl_window.scene`, 124
- `moderngl_window.timers.base`, 119
- `moderngl_window.timers.clock`, 120



## INDEX

### Symbols

`__init__()` (`moderngl_window.conf.Settings` method), 21  
`__init__()` (`moderngl_window.context.base.window.BaseWindow` method), 32  
`__init__()` (`moderngl_window.context.base.window.WindowConfig` method), 28  
`__init__()` (`moderngl_window.context.glfw.window.Window` method), 38  
`__init__()` (`moderngl_window.context.headless.window.Window` method), 45  
`__init__()` (`moderngl_window.context.pyglet.window.Window` method), 51  
`__init__()` (`moderngl_window.context.pyqt5.window.Window` method), 58  
`__init__()` (`moderngl_window.context.pyside2.window.Window` method), 65  
`__init__()` (`moderngl_window.context.sdl2.window.Window` method), 71  
`__init__()` (`moderngl_window.finders.base.BaseFilesystemFinder` method), 105  
`__init__()` (`moderngl_window.finders.data.FilesystemFinder` method), 107  
`__init__()` (`moderngl_window.finders.program.FilesystemFinder` method), 106  
`__init__()` (`moderngl_window.finders.scene.FilesystemFinder` method), 106  
`__init__()` (`moderngl_window.finders.texture.FilesystemFinder` method), 105  
`__init__()` (`moderngl_window.loaders.base.BaseLoader` method), 81  
`__init__()` (`moderngl_window.loaders.data.binary.Loader` method), 93  
`__init__()` (`moderngl_window.loaders.data.json.Loader` method), 91  
`__init__()` (`moderngl_window.loaders.data.text.Loader` method), 92  
`__init__()` (`moderngl_window.loaders.program.separate.Loader` method), 85  
`__init__()` (`moderngl_window.loaders.program.single.Loader` method), 83  
`__init__()` (`moderngl_window.loaders.scene.gltf2.Loader` method), 88  
`__init__()` (`moderngl_window.loaders.scene.stl.Loader` method), 90  
`__init__()` (`moderngl_window.loaders.scene.wavefront.Loader` method), 87  
`__init__()` (`moderngl_window.loaders.texture.array.Loader` method), 86  
`__init__()` (`moderngl_window.loaders.texture.t2d.Loader` method), 82  
`__init__()` (`moderngl_window.meta.base.ResourceDescription` method), 95  
`__init__()` (`moderngl_window.meta.data.DataDescription` method), 103  
`__init__()` (`moderngl_window.meta.program.ProgramDescription` method), 99  
`__init__()` (`moderngl_window.meta.scene.SceneDescription` method), 101  
`__init__()` (`moderngl_window.meta.texture.TextureDescription` method), 96  
`__init__()` (`moderngl_window.opengl.projection.Projection3D` method), 109  
`__init__()` (`moderngl_window.opengl.vao.VAO` method), 111  
`__init__()` (`moderngl_window.resources.base.BaseRegistry` method), 113  
`__init__()` (`moderngl_window.resources.data.DataFiles` method), 117  
`__init__()` (`moderngl_window.resources.programs.Programs` method), 115  
`__init__()` (`moderngl_window.resources.scenes.Scenes` method), 116  
`__init__()` (`moderngl_window.resources.textures.Textures` method), 114  
`__init__()` (`moderngl_window.scene.Camera` method), 121  
`__init__()` (`moderngl_window.scene.KeyboardCamera` method), 122  
`__init__()` (`moderngl_window.scene.Material` method), 128  
`__init__()` (`moderngl_window.scene.MaterialTexture` method), 129  
`__init__()` (`moderngl_window.scene.Mesh` method),

127  
 \_\_init\_\_() (moderngl\_window.scene.MeshProgram  
 method), 129  
 \_\_init\_\_() (moderngl\_window.scene.Node method),  
 126  
 \_\_init\_\_() (moderngl\_window.scene.Scene method),  
 125  
 \_\_init\_\_() (moderngl\_window.timers.base.BaseTimer  
 method), 119  
 \_\_init\_\_() (moderngl\_window.timers.clock.Timer  
 method), 120

## A

activate\_context() (in module mod-  
 erngl\_window), 19  
 add() (moderngl\_window.resources.base.BaseRegistry  
 method), 113  
 add() (moderngl\_window.resources.data.DataFiles  
 method), 117  
 add() (moderngl\_window.resources.programs.Programs  
 method), 115  
 add() (moderngl\_window.resources.scenes.Scenes  
 method), 116  
 add() (moderngl\_window.resources.textures.Textures  
 method), 114  
 add\_attribute() (moderngl\_window.scene.Mesh  
 method), 128  
 add\_child() (moderngl\_window.scene.Node  
 method), 126  
 anisotropy (moderngl\_window.meta.texture.TextureDescription  
 attribute), 97  
 apply() (moderngl\_window.scene.MeshProgram  
 method), 130  
 apply\_default\_settings() (mod-  
 erngl\_window.conf.Settings method), 21  
 apply\_from\_cls() (moderngl\_window.conf.Settings  
 method), 22  
 apply\_from\_dict() (mod-  
 erngl\_window.conf.Settings method), 22  
 apply\_from\_iterable() (mod-  
 erngl\_window.conf.Settings method), 22  
 apply\_from\_module() (mod-  
 erngl\_window.conf.Settings method), 22  
 apply\_from\_module\_name() (mod-  
 erngl\_window.conf.Settings method), 22  
 apply\_mesh\_programs() (mod-  
 erngl\_window.scene.Scene method), 125  
 apply\_settings\_from\_env() (mod-  
 erngl\_window.conf.Settings method), 21  
 aspect\_ratio (mod-  
 erngl\_window.context.base.window.BaseWindow  
 attribute), 36  
 aspect\_ratio (mod-  
 erngl\_window.context.base.window.WindowConfig

attribute), 32  
 aspect\_ratio (mod-  
 erngl\_window.context.glfw.window.Window  
 attribute), 41  
 aspect\_ratio (mod-  
 erngl\_window.context.headless.window.Window  
 attribute), 48  
 aspect\_ratio (mod-  
 erngl\_window.context.pyglet.window.Window  
 attribute), 56  
 aspect\_ratio (mod-  
 erngl\_window.context.pyqt5.window.Window  
 attribute), 62  
 aspect\_ratio (mod-  
 erngl\_window.context.pyside2.window.Window  
 attribute), 69  
 aspect\_ratio (mod-  
 erngl\_window.context.sdl2.window.Window  
 attribute), 75  
 aspect\_ratio (mod-  
 erngl\_window.opengl.projection.Projection3D  
 attribute), 109  
 attr\_names (moderngl\_window.meta.scene.SceneDescription  
 attribute), 101  
 attrs (moderngl\_window.meta.base.ResourceDescription  
 attribute), 95  
 attrs (moderngl\_window.meta.data.DataDescription  
 attribute), 103  
 attrs (moderngl\_window.meta.program.ProgramDescription  
 attribute), 100  
 attrs (moderngl\_window.meta.scene.SceneDescription  
 attribute), 101  
 attrs (moderngl\_window.meta.texture.TextureDescription  
 attribute), 97

## B

BaseFilesystemFinder (in module mod-  
 erngl\_window.finders.base), 105  
 BaseRegistry (in module mod-  
 erngl\_window.resources.base), 113  
 BaseTimer (in module moderngl\_window.timers.base),  
 119  
 bbox() (in module moderngl\_window.geometry), 79  
 buffer() (moderngl\_window.opengl.vao.VAO  
 method), 112  
 buffer\_height (mod-  
 erngl\_window.context.base.window.BaseWindow  
 attribute), 35  
 buffer\_height (mod-  
 erngl\_window.context.glfw.window.Window  
 attribute), 40  
 buffer\_height (mod-  
 erngl\_window.context.headless.window.Window  
 attribute), 47





- [attribute](#)), 63
  - [close\\_func \(moderngl\\_window.context.pyside2.window.Window attribute\)](#), 70
  - [close\\_func \(moderngl\\_window.context.sdl2.window.Window attribute\)](#), 76
  - [color \(moderngl\\_window.scene.Material attribute\)](#), 128
  - [config \(moderngl\\_window.context.base.window.BaseWindow attribute\)](#), 35
  - [config \(moderngl\\_window.context.glfw.window.Window attribute\)](#), 41
  - [config \(moderngl\\_window.context.headless.window.Window attribute\)](#), 48
  - [config \(moderngl\\_window.context.pyglet.window.Window attribute\)](#), 55
  - [config \(moderngl\\_window.context.pyqt5.window.Window attribute\)](#), 62
  - [config \(moderngl\\_window.context.pyside2.window.Window attribute\)](#), 69
  - [config \(moderngl\\_window.context.sdl2.window.Window attribute\)](#), 75
  - [count \(moderngl\\_window.resources.base.BaseRegistry attribute\)](#), 114
  - [count \(moderngl\\_window.resources.data.DataFiles attribute\)](#), 117
  - [count \(moderngl\\_window.resources.programs.Programs attribute\)](#), 116
  - [count \(moderngl\\_window.resources.scenes.Scenes attribute\)](#), 117
  - [count \(moderngl\\_window.resources.textures.Textures attribute\)](#), 115
  - [create\\_window\\_from\\_settings\(\) \(in module moderngl\\_window\)](#), 19
  - [ctx \(moderngl\\_window.context.base.window.BaseWindow attribute\)](#), 34
  - [ctx \(moderngl\\_window.context.glfw.window.Window attribute\)](#), 39
  - [ctx \(moderngl\\_window.context.headless.window.Window attribute\)](#), 46
  - [ctx \(moderngl\\_window.context.pyglet.window.Window attribute\)](#), 54
  - [ctx \(moderngl\\_window.context.pyqt5.window.Window attribute\)](#), 60
  - [ctx \(moderngl\\_window.context.pyside2.window.Window attribute\)](#), 67
  - [ctx \(moderngl\\_window.context.sdl2.window.Window attribute\)](#), 73
  - [ctx \(moderngl\\_window.loaders.base.BaseLoader attribute\)](#), 82
  - [ctx \(moderngl\\_window.loaders.data.binary.Loader attribute\)](#), 93
  - [ctx \(moderngl\\_window.loaders.data.json.Loader attribute\)](#), 91
  - [ctx \(moderngl\\_window.loaders.data.text.Loader attribute\)](#), 92
  - [ctx \(moderngl\\_window.loaders.program.separate.Loader attribute\)](#), 86
  - [ctx \(moderngl\\_window.loaders.program.single.Loader attribute\)](#), 85
  - [ctx \(moderngl\\_window.loaders.scene.gltf2.Loader attribute\)](#), 89
  - [ctx \(moderngl\\_window.loaders.scene.stl.Loader attribute\)](#), 90
  - [ctx \(moderngl\\_window.loaders.scene.wavefront.Loader attribute\)](#), 88
  - [ctx \(moderngl\\_window.loaders.texture.array.Loader attribute\)](#), 87
  - [ctx \(moderngl\\_window.loaders.texture.t2d.Loader attribute\)](#), 83
  - [ctx \(moderngl\\_window.opengl.vao.VAO attribute\)](#), 112
  - [ctx \(moderngl\\_window.scene.MeshProgram attribute\)](#), 130
  - [ctx \(moderngl\\_window.scene.Scene attribute\)](#), 125
  - [ctx \(\) \(in module moderngl\\_window\)](#), 19
  - [cube \(\) \(in module moderngl\\_window.geometry\)](#), 80
  - [cursor \(moderngl\\_window.context.base.window.BaseWindow attribute\)](#), 36
  - [cursor \(moderngl\\_window.context.base.window.WindowConfig attribute\)](#), 32
  - [cursor \(moderngl\\_window.context.glfw.window.Window attribute\)](#), 42
  - [cursor \(moderngl\\_window.context.headless.window.Window attribute\)](#), 49
  - [cursor \(moderngl\\_window.context.pyglet.window.Window attribute\)](#), 56
  - [cursor \(moderngl\\_window.context.pyqt5.window.Window attribute\)](#), 63
  - [cursor \(moderngl\\_window.context.pyside2.window.Window attribute\)](#), 69
  - [cursor \(moderngl\\_window.context.sdl2.window.Window attribute\)](#), 75
- ## D
- [DATA\\_DIRS \(moderngl\\_window.conf.Settings attribute\)](#), 24
  - [DATA\\_FINDERS \(moderngl\\_window.conf.Settings attribute\)](#), 24
  - [DATA\\_LOADERS \(moderngl\\_window.conf.Settings attribute\)](#), 25
  - [DataDescription \(in module moderngl\\_window.meta.data\)](#), 102
  - [DataFiles \(in module moderngl\\_window.resources.data\)](#), 117
  - [default\\_kind \(moderngl\\_window.meta.base.ResourceDescription attribute\)](#), 96
  - [default\\_kind \(moderngl\\_window.meta.data.DataDescription attribute\)](#), 96

[attribute](#)), 103  
 default\_kind (moderngl\_window.meta.program.ProgramDescription [attribute](#)), 100  
 default\_kind (moderngl\_window.meta.scene.SceneDescription [attribute](#)), 102  
 default\_kind (moderngl\_window.meta.texture.TextureDescription [attribute](#)), 98  
 destroy() (moderngl\_window.context.base.window.BaseWindow [method](#)), 33  
 destroy() (moderngl\_window.context.glfw.window.Window [method](#)), 39  
 destroy() (moderngl\_window.context.headless.window.Window [method](#)), 46  
 destroy() (moderngl\_window.context.pyglet.window.Window [method](#)), 52  
 destroy() (moderngl\_window.context.pyqt5.window.Window [method](#)), 59  
 destroy() (moderngl\_window.context.pyside2.window.Window [method](#)), 66  
 destroy() (moderngl\_window.context.sdl2.window.Window [method](#)), 72  
 destroy() (moderngl\_window.scene.Scene [method](#)), 125  
 double\_sided (moderngl\_window.scene.Material [attribute](#)), 129  
 draw() (moderngl\_window.scene.Mesh [method](#)), 127  
 draw() (moderngl\_window.scene.MeshProgram [method](#)), 129  
 draw() (moderngl\_window.scene.Node [method](#)), 126  
 draw() (moderngl\_window.scene.Scene [method](#)), 125  
 draw\_bbox() (moderngl\_window.scene.Mesh [method](#)), 127  
 draw\_bbox() (moderngl\_window.scene.Node [method](#)), 126  
 draw\_bbox() (moderngl\_window.scene.Scene [method](#)), 125

## F

far (moderngl\_window.opengl.projection.Projection3D [attribute](#)), 110  
 fbo (moderngl\_window.context.base.window.BaseWindow [attribute](#)), 34  
 fbo (moderngl\_window.context.glfw.window.Window [attribute](#)), 39  
 fbo (moderngl\_window.context.headless.window.Window [attribute](#)), 47  
 fbo (moderngl\_window.context.pyglet.window.Window [attribute](#)), 54  
 fbo (moderngl\_window.context.pyqt5.window.Window [attribute](#)), 60  
 fbo (moderngl\_window.context.pyside2.window.Window [attribute](#)), 67  
 fbo (moderngl\_window.context.sdl2.window.Window [attribute](#)), 73  
 file\_extensions (moderngl\_window.loaders.base.BaseLoader [attribute](#)), 82  
 file\_extensions (moderngl\_window.loaders.data.binary.Loader [attribute](#)), 93  
 file\_extensions (moderngl\_window.loaders.data.json.Loader [attribute](#)), 91  
 file\_extensions (moderngl\_window.loaders.data.text.Loader [attribute](#)), 92  
 file\_extensions (moderngl\_window.loaders.program.separate.Loader [attribute](#)), 86  
 file\_extensions (moderngl\_window.loaders.program.single.Loader [attribute](#)), 84  
 file\_extensions (moderngl\_window.loaders.scene.gltf2.Loader [attribute](#)), 89  
 file\_extensions (moderngl\_window.loaders.scene.stl.Loader [attribute](#)), 90  
 file\_extensions (moderngl\_window.loaders.scene.wavefront.Loader [attribute](#)), 88  
 file\_extensions (moderngl\_window.loaders.texture.array.Loader [attribute](#)), 87  
 file\_extensions (moderngl\_window.loaders.texture.t2d.Loader [attribute](#)), 83  
 FilesystemFinder (in module moderngl\_window.finders.data), 107  
 FilesystemFinder (in module moderngl\_window.finders.program), 106  
 FilesystemFinder (in module moderngl\_window.finders.scene), 106  
 FilesystemFinder (in module moderngl\_window.finders.texture), 105  
 find() (moderngl\_window.finders.base.BaseFilesystemFinder [method](#)), 105  
 find() (moderngl\_window.finders.data.FilesystemFinder [method](#)), 107  
 find() (moderngl\_window.finders.program.FilesystemFinder [method](#)), 106  
 find() (moderngl\_window.finders.scene.FilesystemFinder [method](#)), 106  
 find() (moderngl\_window.finders.texture.FilesystemFinder [method](#)), 105

<i>method</i> ), 105	<i>erngl_window.loaders.texture.t2d.Loader</i>
<i>find_data()</i> ( <i>moderngl_window.loaders.base.BaseLoader</i>	<i>method</i> ), 82
<i>method</i> ), 81	<i>find_scene()</i> ( <i>mod-</i>
<i>find_data()</i> ( <i>moderngl_window.loaders.data.binary.Loader</i>	<i>erngl_window.loaders.base.BaseLoader</i>
<i>method</i> ), 93	<i>method</i> ), 81
<i>find_data()</i> ( <i>moderngl_window.loaders.data.json.Loader</i>	<i>find_scene()</i> ( <i>mod-</i>
<i>method</i> ), 91	<i>erngl_window.loaders.data.binary.Loader</i>
<i>find_data()</i> ( <i>moderngl_window.loaders.data.text.Loader</i>	<i>method</i> ), 93
<i>method</i> ), 92	<i>find_scene()</i> ( <i>mod-</i>
<i>find_data()</i> ( <i>moderngl_window.loaders.program.separate.Loader</i>	<i>erngl_window.loaders.data.json.Loader</i>
<i>method</i> ), 85	<i>method</i> ), 91
<i>find_data()</i> ( <i>moderngl_window.loaders.program.single.Loader</i>	<i>find_scene()</i> ( <i>mod-</i>
<i>method</i> ), 84	<i>erngl_window.loaders.data.text.Loader</i>
<i>find_data()</i> ( <i>moderngl_window.loaders.scene.gltf2.Loader</i>	<i>method</i> ), 92
<i>method</i> ), 88	<i>find_scene()</i> ( <i>mod-</i>
<i>find_data()</i> ( <i>moderngl_window.loaders.scene.stl.Loader</i>	<i>erngl_window.loaders.program.separate.Loader</i>
<i>method</i> ), 90	<i>method</i> ), 85
<i>find_data()</i> ( <i>moderngl_window.loaders.scene.wavefront.Loader</i>	<i>find_scene()</i> ( <i>mod-</i>
<i>method</i> ), 87	<i>erngl_window.loaders.program.single.Loader</i>
<i>find_data()</i> ( <i>moderngl_window.loaders.texture.array.Loader</i>	<i>method</i> ), 84
<i>method</i> ), 86	<i>find_scene()</i> ( <i>mod-</i>
<i>find_data()</i> ( <i>moderngl_window.loaders.texture.t2d.Loader</i>	<i>erngl_window.loaders.scene.gltf2.Loader</i>
<i>method</i> ), 82	<i>method</i> ), 89
<i>find_program()</i> ( <i>mod-</i>	<i>find_scene()</i> ( <i>mod-</i>
<i>erngl_window.loaders.base.BaseLoader</i>	<i>erngl_window.loaders.scene.stl.Loader</i>
<i>method</i> ), 81	<i>method</i> ), 90
<i>find_program()</i> ( <i>mod-</i>	<i>find_scene()</i> ( <i>mod-</i>
<i>erngl_window.loaders.data.binary.Loader</i>	<i>erngl_window.loaders.scene.wavefront.Loader</i>
<i>method</i> ), 93	<i>method</i> ), 88
<i>find_program()</i> ( <i>mod-</i>	<i>find_scene()</i> ( <i>mod-</i>
<i>erngl_window.loaders.data.json.Loader</i>	<i>erngl_window.loaders.texture.array.Loader</i>
<i>method</i> ), 91	<i>method</i> ), 86
<i>find_program()</i> ( <i>mod-</i>	<i>find_scene()</i> ( <i>mod-</i>
<i>erngl_window.loaders.data.text.Loader</i>	<i>erngl_window.loaders.texture.t2d.Loader</i>
<i>method</i> ), 92	<i>method</i> ), 83
<i>find_program()</i> ( <i>mod-</i>	<i>find_texture()</i> ( <i>mod-</i>
<i>erngl_window.loaders.program.separate.Loader</i>	<i>erngl_window.loaders.base.BaseLoader</i>
<i>method</i> ), 85	<i>method</i> ), 81
<i>find_program()</i> ( <i>mod-</i>	<i>find_texture()</i> ( <i>mod-</i>
<i>erngl_window.loaders.program.single.Loader</i>	<i>erngl_window.loaders.data.binary.Loader</i>
<i>method</i> ), 84	<i>method</i> ), 93
<i>find_program()</i> ( <i>mod-</i>	<i>find_texture()</i> ( <i>mod-</i>
<i>erngl_window.loaders.scene.gltf2.Loader</i>	<i>erngl_window.loaders.data.json.Loader</i>
<i>method</i> ), 88	<i>method</i> ), 91
<i>find_program()</i> ( <i>mod-</i>	<i>find_texture()</i> ( <i>mod-</i>
<i>erngl_window.loaders.scene.stl.Loader</i>	<i>erngl_window.loaders.data.text.Loader</i>
<i>method</i> ), 90	<i>method</i> ), 92
<i>find_program()</i> ( <i>mod-</i>	<i>find_texture()</i> ( <i>mod-</i>
<i>erngl_window.loaders.scene.wavefront.Loader</i>	<i>erngl_window.loaders.program.separate.Loader</i>
<i>method</i> ), 87	<i>method</i> ), 85
<i>find_program()</i> ( <i>mod-</i>	<i>find_texture()</i> ( <i>mod-</i>
<i>erngl_window.loaders.texture.array.Loader</i>	<i>erngl_window.loaders.program.single.Loader</i>
<i>method</i> ), 86	<i>method</i> ), 84
<i>find_program()</i> ( <i>mod-</i>	<i>find_texture()</i> ( <i>mod-</i>

- [erngl\\_window.loaders.scene.gltf2.Loader method](#)), 88
  - [find\\_texture\(\)](#) ([moderngl\\_window.loaders.scene.stl.Loader method](#)), 90
  - [find\\_texture\(\)](#) ([moderngl\\_window.loaders.scene.wavefront.Loader method](#)), 87
  - [find\\_texture\(\)](#) ([moderngl\\_window.loaders.texture.array.Loader method](#)), 86
  - [find\\_texture\(\)](#) ([moderngl\\_window.loaders.texture.t2d.Loader method](#)), 83
  - [find\\_window\\_classes\(\)](#) (in module [moderngl\\_window](#)), 19
  - [fixed\\_aspect\\_ratio](#) ([moderngl\\_window.context.base.window.BaseWindow attribute](#)), 36
  - [fixed\\_aspect\\_ratio](#) ([moderngl\\_window.context.glfw.window.Window attribute](#)), 42
  - [fixed\\_aspect\\_ratio](#) ([moderngl\\_window.context.headless.window.Window attribute](#)), 49
  - [fixed\\_aspect\\_ratio](#) ([moderngl\\_window.context.pyglet.window.Window attribute](#)), 56
  - [fixed\\_aspect\\_ratio](#) ([moderngl\\_window.context.pyqt5.window.Window attribute](#)), 62
  - [fixed\\_aspect\\_ratio](#) ([moderngl\\_window.context.pyside2.window.Window attribute](#)), 69
  - [fixed\\_aspect\\_ratio](#) ([moderngl\\_window.context.sdl2.window.Window attribute](#)), 75
  - [flip](#) ([moderngl\\_window.meta.texture.TextureDescription attribute](#)), 97
  - [fov](#) ([moderngl\\_window.opengl.projection.Projection3D attribute](#)), 110
  - [fragment\\_shader](#) ([moderngl\\_window.meta.program.ProgramDescription attribute](#)), 99
  - [frames](#) ([moderngl\\_window.context.base.window.BaseWindow attribute](#)), 35
  - [frames](#) ([moderngl\\_window.context.glfw.window.Window attribute](#)), 41
  - [frames](#) ([moderngl\\_window.context.headless.window.Window attribute](#)), 48
  - [frames](#) ([moderngl\\_window.context.pyglet.window.Window attribute](#)), 55
  - [frames](#) ([moderngl\\_window.context.pyqt5.window.Window attribute](#)), 62
  - [frames](#) ([moderngl\\_window.context.pyside2.window.Window attribute](#)), 69
  - [frames](#) ([moderngl\\_window.context.sdl2.window.Window attribute](#)), 74
  - [fullscreen](#) ([moderngl\\_window.context.base.window.BaseWindow attribute](#)), 35
  - [fullscreen](#) ([moderngl\\_window.context.glfw.window.Window attribute](#)), 41
  - [fullscreen](#) ([moderngl\\_window.context.headless.window.Window attribute](#)), 48
  - [fullscreen](#) ([moderngl\\_window.context.pyglet.window.Window attribute](#)), 55
  - [fullscreen](#) ([moderngl\\_window.context.pyqt5.window.Window attribute](#)), 62
  - [fullscreen](#) ([moderngl\\_window.context.pyside2.window.Window attribute](#)), 69
  - [fullscreen](#) ([moderngl\\_window.context.sdl2.window.Window attribute](#)), 75
- ## G
- [geometry\\_shader](#) ([moderngl\\_window.meta.program.ProgramDescription attribute](#)), 99
  - [get\\_buffer\\_by\\_name\(\)](#) ([moderngl\\_window.opengl.vao.VAO method](#)), 112
  - [get\\_local\\_window\\_cls\(\)](#) (in module [moderngl\\_window](#)), 19
  - [get\\_window\\_cls\(\)](#) (in module [moderngl\\_window](#)), 19
  - [gl\\_version](#) ([moderngl\\_window.context.base.window.BaseWindow attribute](#)), 34
  - [gl\\_version](#) ([moderngl\\_window.context.base.window.WindowConfig attribute](#)), 31
  - [gl\\_version](#) ([moderngl\\_window.context.glfw.window.Window attribute](#)), 40
  - [gl\\_version](#) ([moderngl\\_window.context.headless.window.Window attribute](#)), 47
  - [gl\\_version](#) ([moderngl\\_window.context.pyglet.window.Window attribute](#)), 54
  - [gl\\_version](#) ([moderngl\\_window.context.pyqt5.window.Window attribute](#)), 61
  - [gl\\_version](#) ([moderngl\\_window.context.pyside2.window.Window attribute](#)), 67
  - [gl\\_version](#) ([moderngl\\_window.context.sdl2.window.Window attribute](#)), 73
  - [gl\\_version\\_code](#) ([moderngl\\_window.context.base.window.BaseWindow attribute](#)), 38
  - [gl\\_version\\_code](#) ([moderngl\\_window.context.glfw.window.Window attribute](#)), 43
  - [gl\\_version\\_code](#) ([moderngl\\_window.context.headless.window.Window](#)



- [attribute](#)), 51
  - [gl\\_version\\_code](#) ([moderngl\\_window.context.pyglet.window.Window](#) attribute), 58
  - [gl\\_version\\_code](#) ([moderngl\\_window.context.pyqt5.window.Window](#) attribute), 64
  - [gl\\_version\\_code](#) ([moderngl\\_window.context.pyside2.window.Window](#) attribute), 71
  - [gl\\_version\\_code](#) ([moderngl\\_window.context.sdl2.window.Window](#) attribute), 77
  - [glfw\\_char\\_callback\(\)](#) ([moderngl\\_window.context.glfw.window.Window](#) method), 45
  - [glfw\\_cursor\\_enter\(\)](#) ([moderngl\\_window.context.glfw.window.Window](#) method), 45
  - [glfw\\_key\\_event\\_callback\(\)](#) ([moderngl\\_window.context.glfw.window.Window](#) method), 44
  - [glfw\\_mouse\\_button\\_callback\(\)](#) ([moderngl\\_window.context.glfw.window.Window](#) method), 44
  - [glfw\\_mouse\\_event\\_callback\(\)](#) ([moderngl\\_window.context.glfw.window.Window](#) method), 44
  - [glfw\\_mouse\\_scroll\\_callback\(\)](#) ([moderngl\\_window.context.glfw.window.Window](#) method), 44
  - [glfw\\_window\\_focus\(\)](#) ([moderngl\\_window.context.glfw.window.Window](#) method), 45
  - [glfw\\_window\\_iconify\(\)](#) ([moderngl\\_window.context.glfw.window.Window](#) method), 45
  - [glfw\\_window\\_resize\\_callback\(\)](#) ([moderngl\\_window.context.glfw.window.Window](#) method), 44
- ## H
- [has\\_normals\(\)](#) ([moderngl\\_window.scene.Mesh](#) method), 128
  - [has\\_uvs\(\)](#) ([moderngl\\_window.scene.Mesh](#) method), 128
  - [height](#) ([moderngl\\_window.context.base.window.BaseWindow](#) attribute), 34
  - [height](#) ([moderngl\\_window.context.glfw.window.Window](#) attribute), 40
  - [height](#) ([moderngl\\_window.context.headless.window.Window](#) attribute), 47
  - [height](#) ([moderngl\\_window.context.pyglet.window.Window](#) attribute), 54
  - [height](#) ([moderngl\\_window.context.pyqt5.window.Window](#) attribute), 61
  - [height](#) ([moderngl\\_window.context.pyside2.window.Window](#) attribute), 67
  - [height](#) ([moderngl\\_window.context.sdl2.window.Window](#) attribute), 73
  - [hide\\_event\(\)](#) ([moderngl\\_window.context.pyqt5.window.Window](#) method), 60
  - [hide\\_event\(\)](#) ([moderngl\\_window.context.pyside2.window.Window](#) method), 67
- ## I
- [iconify\\_func](#) ([moderngl\\_window.context.base.window.BaseWindow](#) attribute), 37
  - [iconify\\_func](#) ([moderngl\\_window.context.glfw.window.Window](#) attribute), 43
  - [iconify\\_func](#) ([moderngl\\_window.context.headless.window.Window](#) attribute), 50
  - [iconify\\_func](#) ([moderngl\\_window.context.pyglet.window.Window](#) attribute), 57
  - [iconify\\_func](#) ([moderngl\\_window.context.pyqt5.window.Window](#) attribute), 63
  - [iconify\\_func](#) ([moderngl\\_window.context.pyside2.window.Window](#) attribute), 70
  - [iconify\\_func](#) ([moderngl\\_window.context.sdl2.window.Window](#) attribute), 76
  - [image](#) ([moderngl\\_window.meta.texture.TextureDescription](#) attribute), 97
  - [index\\_buffer\(\)](#) ([moderngl\\_window.opengl.vao.VAO](#) method), 112
  - [init\\_mgl\\_context\(\)](#) ([moderngl\\_window.context.base.window.BaseWindow](#) method), 33
  - [init\\_mgl\\_context\(\)](#) ([moderngl\\_window.context.glfw.window.Window](#) method), 38
  - [init\\_mgl\\_context\(\)](#) ([moderngl\\_window.context.headless.window.Window](#) method), 46
  - [init\\_mgl\\_context\(\)](#) ([moderngl\\_window.context.pyglet.window.Window](#) method), 51
  - [init\\_mgl\\_context\(\)](#) ([moderngl\\_window.context.pyqt5.window.Window](#) method), 58

- method*), 58
- `init_mgl_context()` (*moderngl\_window.context.pyside2.window.Window* *method*), 65
- `init_mgl_context()` (*moderngl\_window.context.sdl2.window.Window* *method*), 72
- `instance()` (*moderngl\_window.opengl.vao.VAO* *method*), 112
- `is_closing` (*moderngl\_window.context.base.window.BaseWindow* *attribute*), 37
- `is_closing` (*moderngl\_window.context.glfw.window.Window* *attribute*), 43
- `is_closing` (*moderngl\_window.context.headless.window.Window* *attribute*), 50
- `is_closing` (*moderngl\_window.context.pyglet.window.Window* *attribute*), 57
- `is_closing` (*moderngl\_window.context.pyqt5.window.Window* *attribute*), 64
- `is_closing` (*moderngl\_window.context.pyside2.window.Window* *attribute*), 71
- `is_closing` (*moderngl\_window.context.sdl2.window.Window* *attribute*), 77
- `is_key_pressed()` (*moderngl\_window.context.base.window.BaseWindow* *method*), 33
- `is_key_pressed()` (*moderngl\_window.context.glfw.window.Window* *method*), 38
- `is_key_pressed()` (*moderngl\_window.context.headless.window.Window* *method*), 46
- `is_key_pressed()` (*moderngl\_window.context.pyglet.window.Window* *method*), 51
- `is_key_pressed()` (*moderngl\_window.context.pyqt5.window.Window* *method*), 58
- `is_key_pressed()` (*moderngl\_window.context.pyside2.window.Window* *method*), 65
- `is_key_pressed()` (*moderngl\_window.context.sdl2.window.Window* *method*), 72
- `is_paused` (*moderngl\_window.timers.base.BaseTimer* *attribute*), 119
- `is_paused` (*moderngl\_window.timers.clock.Timer* *attribute*), 120
- `is_running` (*moderngl\_window.timers.base.BaseTimer* *attribute*), 119
- `is_running` (*moderngl\_window.timers.clock.Timer* *attribute*), 120
- ## K
- `key_event()` (*moderngl\_window.context.base.window.WindowConfig* *method*), 28
- `key_event_func` (*moderngl\_window.context.base.window.BaseWindow* *attribute*), 37
- `key_event_func` (*moderngl\_window.context.glfw.window.Window* *attribute*), 43
- `key_event_func` (*moderngl\_window.context.headless.window.Window* *attribute*), 50
- `key_event_func` (*moderngl\_window.context.pyglet.window.Window* *attribute*), 57
- `key_event_func` (*moderngl\_window.context.pyqt5.window.Window* *attribute*), 63
- `key_event_func` (*moderngl\_window.context.pyside2.window.Window* *attribute*), 70
- `key_event_func` (*moderngl\_window.context.sdl2.window.Window* *attribute*), 76
- `key_input()` (*moderngl\_window.scene.KeyboardCamera* *method*), 123
- `key_pressed_event()` (*moderngl\_window.context.pyqt5.window.Window* *method*), 60
- `key_pressed_event()` (*moderngl\_window.context.pyside2.window.Window* *method*), 66
- `key_release_event()` (*moderngl\_window.context.pyqt5.window.Window* *method*), 59
- `key_release_event()` (*moderngl\_window.context.pyside2.window.Window* *method*), 66
- `KeyboardCamera` (in module *moderngl\_window.scene*), 122
- `keys` (*moderngl\_window.context.base.window.BaseWindow* *attribute*), 34
- `keys` (*moderngl\_window.context.glfw.window.Window* *attribute*), 39
- `keys` (*moderngl\_window.context.headless.window.Window* *attribute*), 46
- `keys` (*moderngl\_window.context.pyglet.window.Window* *attribute*), 54
- `keys` (*moderngl\_window.context.pyqt5.window.Window* *attribute*), 60
- `keys` (*moderngl\_window.context.pyside2.window.Window* *attribute*), 67
- `keys` (*moderngl\_window.context.sdl2.window.Window* *attribute*), 73

kind (moderngl_window.loaders.base.BaseLoader attribute), 82	load () (moderngl_window.loaders.program.separate.Loader method), 85
kind (moderngl_window.loaders.data.binary.Loader attribute), 93	load () (moderngl_window.loaders.program.single.Loader method), 83
kind (moderngl_window.loaders.data.json.Loader attribute), 91	load () (moderngl_window.loaders.scene.gltf2.Loader method), 88
kind (moderngl_window.loaders.data.text.Loader attribute), 92	load () (moderngl_window.loaders.scene.stl.Loader method), 90
kind (moderngl_window.loaders.program.separate.Loader attribute), 86	load () (moderngl_window.loaders.scene.wavefront.Loader method), 87
kind (moderngl_window.loaders.program.single.Loader attribute), 84	load () (moderngl_window.loaders.texture.array.Loader method), 86
kind (moderngl_window.loaders.scene.gltf2.Loader attribute), 89	load () (moderngl_window.loaders.texture.t2d.Loader method), 82
kind (moderngl_window.loaders.scene.stl.Loader attribute), 90	load () (moderngl_window.resources.base.BaseRegistry method), 113
kind (moderngl_window.loaders.scene.wavefront.Loader attribute), 88	load () (moderngl_window.resources.data.DataFiles method), 117
kind (moderngl_window.loaders.texture.array.Loader attribute), 87	load () (moderngl_window.resources.programs.Programs method), 115
kind (moderngl_window.loaders.texture.t2d.Loader attribute), 83	load () (moderngl_window.resources.scenes.Scenes method), 116
kind (moderngl_window.meta.base.ResourceDescription attribute), 95	load () (moderngl_window.resources.textures.Textures method), 114
kind (moderngl_window.meta.data.DataDescription attribute), 103	load_binary () (moderngl_window.context.base.window.WindowConfig method), 31
kind (moderngl_window.meta.program.ProgramDescription attribute), 100	load_glb () (moderngl_window.loaders.scene.gltf2.Loader method), 89
kind (moderngl_window.meta.scene.SceneDescription attribute), 102	load_gltf () (moderngl_window.loaders.scene.gltf2.Loader method), 89
kind (moderngl_window.meta.texture.TextureDescription attribute), 98	load_images () (moderngl_window.loaders.scene.gltf2.Loader method), 89
<b>L</b>	
label (moderngl_window.meta.base.ResourceDescription attribute), 95	load_json () (moderngl_window.context.base.window.WindowConfig method), 30
label (moderngl_window.meta.data.DataDescription attribute), 103	load_materials () (moderngl_window.loaders.scene.gltf2.Loader method), 89
label (moderngl_window.meta.program.ProgramDescription attribute), 100	load_meshes () (moderngl_window.loaders.scene.gltf2.Loader method), 89
label (moderngl_window.meta.scene.SceneDescription attribute), 102	load_node () (moderngl_window.loaders.scene.gltf2.Loader method), 89
label (moderngl_window.meta.texture.TextureDescription attribute), 97	load_nodes () (moderngl_window.loaders.scene.gltf2.Loader method), 89
layers (moderngl_window.meta.texture.TextureDescription attribute), 97	load_pool () (moderngl_window.resources.base.BaseRegistry method), 113
load () (moderngl_window.loaders.base.BaseLoader method), 81	load_pool () (moderngl_window.resources.data.DataFiles method), 117
load () (moderngl_window.loaders.data.binary.Loader method), 93	load_pool () (moderngl_window.resources.programs.Programs method), 115
load () (moderngl_window.loaders.data.json.Loader method), 91	load_pool () (moderngl_window.resources.scenes.Scenes method), 116
load () (moderngl_window.loaders.data.text.Loader method), 92	



method), 116

load\_pool() (moderngl\_window.resources.textures.Textures method), 114

load\_program() (moderngl\_window.context.base.window.WindowConfig method), 30

load\_samplers() (moderngl\_window.loaders.scene.glTF2.Loader method), 89

load\_scene() (moderngl\_window.context.base.window.WindowConfig method), 31

load\_shader() (moderngl\_window.loaders.program.separate.Loader method), 86

load\_text() (moderngl\_window.context.base.window.WindowConfig method), 30

load\_texture\_2d() (moderngl\_window.context.base.window.WindowConfig method), 29

load\_texture\_array() (moderngl\_window.context.base.window.WindowConfig method), 29

load\_textures() (moderngl\_window.loaders.scene.glTF2.Loader method), 89

loader\_cls (moderngl\_window.meta.base.ResourceDescription attribute), 96

loader\_cls (moderngl\_window.meta.data.DataDescription attribute), 103

loader\_cls (moderngl\_window.meta.program.ProgramDescription attribute), 100

loader\_cls (moderngl\_window.meta.scene.SceneDescription attribute), 102

loader\_cls (moderngl\_window.meta.texture.TextureDescription attribute), 98

loaders (moderngl\_window.resources.base.BaseRegistry attribute), 114

loaders (moderngl\_window.resources.data.DataFiles attribute), 118

loaders (moderngl\_window.resources.programs.Programs attribute), 116

loaders (moderngl\_window.resources.scenes.Scenes attribute), 117

loaders (moderngl\_window.resources.textures.Textures attribute), 115

log\_level (moderngl\_window.context.base.window.WindowConfig attribute), 32

look\_at() (moderngl\_window.scene.Camera method), 121

look\_at() (moderngl\_window.scene.KeyboardCamera method), 123

**M**

mat\_texture (moderngl\_window.scene.Material attribute), 128

Material (in module moderngl\_window.scene), 128

MaterialTexture (in module moderngl\_window.scene), 129

matrix (moderngl\_window.opengl.projection.Projection3D attribute), 110

matrix (moderngl\_window.scene.Camera attribute), 122

matrix (moderngl\_window.scene.KeyboardCamera attribute), 124

Mesh (in module moderngl\_window.scene), 127

MeshProgram (in module moderngl\_window.scene), 129

mipmap (moderngl\_window.meta.texture.TextureDescription attribute), 97

mipmap\_levels (moderngl\_window.meta.texture.TextureDescription attribute), 97

model\_matrix (moderngl\_window.scene.Scene attribute), 125

moderngl\_window (module), 17

moderngl\_window.conf (module), 20

moderngl\_window.context.base.window (module), 27, 32

moderngl\_window.context.glfw.window (module), 38

moderngl\_window.context.headless.window (module), 45

moderngl\_window.context.pyglet.window (module), 51

moderngl\_window.context.pyqt5.window (module), 58

moderngl\_window.context.pyside2.window (module), 64

moderngl\_window.context.sdl2.window (module), 71

moderngl\_window.finders.base (module), 105

moderngl\_window.finders.data (module), 106

moderngl\_window.finders.program (module), 106

moderngl\_window.finders.scene (module), 106

moderngl\_window.finders.texture (module), 105

moderngl\_window.geometry (module), 77

moderngl\_window.loaders.base (module), 81

moderngl\_window.loaders.data.binary (module), 92

moderngl\_window.loaders.data.json (module), 90

moderngl\_window.loaders.data.text (module), 91

[moderngl\\_window.loaders.program.separate \(module\)](#), 85  
[moderngl\\_window.loaders.program.single \(module\)](#), 83  
[moderngl\\_window.loaders.scene.gltf2 \(module\)](#), 88  
[moderngl\\_window.loaders.scene.stl \(module\)](#), 89  
[moderngl\\_window.loaders.scene.wavefront \(module\)](#), 87  
[moderngl\\_window.loaders.texture.array \(module\)](#), 86  
[moderngl\\_window.loaders.texture.t2d \(module\)](#), 82  
[moderngl\\_window.meta.base \(module\)](#), 95  
[moderngl\\_window.meta.data \(module\)](#), 102  
[moderngl\\_window.meta.program \(module\)](#), 98  
[moderngl\\_window.meta.scene \(module\)](#), 100  
[moderngl\\_window.meta.texture \(module\)](#), 96  
[moderngl\\_window.opengl.projection \(module\)](#), 109  
[moderngl\\_window.opengl.vao \(module\)](#), 110  
[moderngl\\_window.resources \(module\)](#), 112  
[moderngl\\_window.resources.base \(module\)](#), 113  
[moderngl\\_window.resources.data \(module\)](#), 117  
[moderngl\\_window.resources.programs \(module\)](#), 115  
[moderngl\\_window.resources.scenes \(module\)](#), 116  
[moderngl\\_window.resources.textures \(module\)](#), 114  
[moderngl\\_window.scene \(module\)](#), 121, 122, 124, 125, 127–129  
[moderngl\\_window.timers.base \(module\)](#), 119  
[moderngl\\_window.timers.clock \(module\)](#), 120  
[modifiers \(moderngl\\_window.context.base.window.BaseWindow attribute\)](#), 38  
[modifiers \(moderngl\\_window.context.glfw.window.Window attribute\)](#), 43  
[modifiers \(moderngl\\_window.context.headless.window.Window attribute\)](#), 50  
[modifiers \(moderngl\\_window.context.pyglet.window.Window attribute\)](#), 58  
[modifiers \(moderngl\\_window.context.pyqt5.window.Window attribute\)](#), 64  
[modifiers \(moderngl\\_window.context.pyside2.window.Window attribute\)](#), 71  
[modifiers \(moderngl\\_window.context.sdl2.window.Window attribute\)](#), 77  
[mouse \(moderngl\\_window.context.base.window.BaseWindow attribute\)](#), 37  
[mouse \(moderngl\\_window.context.glfw.window.Window attribute\)](#), 43  
[mouse \(moderngl\\_window.context.headless.window.Window attribute\)](#), 50  
[mouse \(moderngl\\_window.context.pyglet.window.Window attribute\)](#), 57  
[mouse \(moderngl\\_window.context.pyqt5.window.Window attribute\)](#), 64  
[mouse \(moderngl\\_window.context.pyside2.window.Window attribute\)](#), 71  
[mouse \(moderngl\\_window.context.sdl2.window.Window attribute\)](#), 77  
[mouse\\_drag\\_event \(\) \(mod-erngl\\_window.context.base.window.WindowConfig method\)](#), 29  
[mouse\\_drag\\_event\\_func \(mod-erngl\\_window.context.base.window.BaseWindow attribute\)](#), 37  
[mouse\\_drag\\_event\\_func \(mod-erngl\\_window.context.glfw.window.Window attribute\)](#), 43  
[mouse\\_drag\\_event\\_func \(mod-erngl\\_window.context.headless.window.Window attribute\)](#), 50  
[mouse\\_drag\\_event\\_func \(mod-erngl\\_window.context.pyglet.window.Window attribute\)](#), 57  
[mouse\\_drag\\_event\\_func \(mod-erngl\\_window.context.pyqt5.window.Window attribute\)](#), 64  
[mouse\\_drag\\_event\\_func \(mod-erngl\\_window.context.pyside2.window.Window attribute\)](#), 70  
[mouse\\_drag\\_event\\_func \(mod-erngl\\_window.context.sdl2.window.Window attribute\)](#), 76  
[mouse\\_exclusivity \(mod-erngl\\_window.context.base.window.BaseWindow attribute\)](#), 36  
[mouse\\_exclusivity \(mod-erngl\\_window.context.glfw.window.Window attribute\)](#), 42  
[mouse\\_exclusivity \(mod-erngl\\_window.context.headless.window.Window attribute\)](#), 49  
[mouse\\_exclusivity \(mod-erngl\\_window.context.pyglet.window.Window attribute\)](#), 56  
[mouse\\_exclusivity \(mod-erngl\\_window.context.pyqt5.window.Window attribute\)](#), 63  
[mouse\\_exclusivity \(mod-erngl\\_window.context.pyside2.window.Window attribute\)](#), 70  
[mouse\\_exclusivity \(mod-erngl\\_window.context.sdl2.window.Window attribute\)](#), 77

<i>erngl_window.context.sdl2.window.Window</i> attribute), 75	<i>erngl_window.context.pyqt5.window.Window</i> attribute), 64
<i>mouse_move_event()</i> (mod- <i>erngl_window.context.pyqt5.window.Window</i> method), 59	<i>mouse_press_event_func</i> (mod- <i>erngl_window.context.pyside2.window.Window</i> attribute), 70
<i>mouse_move_event()</i> (mod- <i>erngl_window.context.pyside2.window.Window</i> method), 66	<i>mouse_press_event_func</i> (mod- <i>erngl_window.context.sdl2.window.Window</i> attribute), 76
<i>mouse_position_event()</i> (mod- <i>erngl_window.context.base.window.WindowConfig</i> method), 28	<i>mouse_release_event()</i> (mod- <i>erngl_window.context.base.window.WindowConfig</i> method), 29
<i>mouse_position_event_func</i> (mod- <i>erngl_window.context.base.window.BaseWindow</i> attribute), 37	<i>mouse_release_event()</i> (mod- <i>erngl_window.context.pyqt5.window.Window</i> method), 59
<i>mouse_position_event_func</i> (mod- <i>erngl_window.context.glfw.window.Window</i> attribute), 43	<i>mouse_release_event()</i> (mod- <i>erngl_window.context.pyside2.window.Window</i> method), 66
<i>mouse_position_event_func</i> (mod- <i>erngl_window.context.headless.window.Window</i> attribute), 50	<i>mouse_release_event_func</i> (mod- <i>erngl_window.context.base.window.BaseWindow</i> attribute), 37
<i>mouse_position_event_func</i> (mod- <i>erngl_window.context.pyglet.window.Window</i> attribute), 57	<i>mouse_release_event_func</i> (mod- <i>erngl_window.context.glfw.window.Window</i> attribute), 43
<i>mouse_position_event_func</i> (mod- <i>erngl_window.context.pyqt5.window.Window</i> attribute), 64	<i>mouse_release_event_func</i> (mod- <i>erngl_window.context.headless.window.Window</i> attribute), 50
<i>mouse_position_event_func</i> (mod- <i>erngl_window.context.pyside2.window.Window</i> attribute), 70	<i>mouse_release_event_func</i> (mod- <i>erngl_window.context.pyglet.window.Window</i> attribute), 57
<i>mouse_position_event_func</i> (mod- <i>erngl_window.context.sdl2.window.Window</i> attribute), 76	<i>mouse_release_event_func</i> (mod- <i>erngl_window.context.pyqt5.window.Window</i> attribute), 64
<i>mouse_press_event()</i> (mod- <i>erngl_window.context.base.window.WindowConfig</i> method), 28	<i>mouse_release_event_func</i> (mod- <i>erngl_window.context.pyside2.window.Window</i> attribute), 70
<i>mouse_press_event()</i> (mod- <i>erngl_window.context.pyqt5.window.Window</i> method), 60	<i>mouse_release_event_func</i> (mod- <i>erngl_window.context.sdl2.window.Window</i> attribute), 76
<i>mouse_press_event()</i> (mod- <i>erngl_window.context.pyside2.window.Window</i> method), 66	<i>mouse_scroll_event()</i> (mod- <i>erngl_window.context.base.window.WindowConfig</i> method), 29
<i>mouse_press_event_func</i> (mod- <i>erngl_window.context.base.window.BaseWindow</i> attribute), 37	<i>mouse_scroll_event_func</i> (mod- <i>erngl_window.context.base.window.BaseWindow</i> attribute), 37
<i>mouse_press_event_func</i> (mod- <i>erngl_window.context.glfw.window.Window</i> attribute), 43	<i>mouse_scroll_event_func</i> (mod- <i>erngl_window.context.glfw.window.Window</i> attribute), 43
<i>mouse_press_event_func</i> (mod- <i>erngl_window.context.headless.window.Window</i> attribute), 50	<i>mouse_scroll_event_func</i> (mod- <i>erngl_window.context.headless.window.Window</i> attribute), 50
<i>mouse_press_event_func</i> (mod- <i>erngl_window.context.pyglet.window.Window</i> attribute), 57	<i>mouse_scroll_event_func</i> (mod- <i>erngl_window.context.pyglet.window.Window</i> attribute), 57
<i>mouse_press_event_func</i> (mod-	<i>mouse_scroll_event_func</i> (mod-

- `erngl_window.context.pyqt5.window.Window`  
`attribute`), 64
  - `mouse_scroll_event_func` (`moderngl_window.context.pyside2.window.Window`  
`attribute`), 71
  - `mouse_scroll_event_func` (`moderngl_window.context.sdl2.window.Window`  
`attribute`), 76
  - `mouse_sensitivity` (`moderngl_window.scene.KeyboardCamera` `attribute`), 124
  - `mouse_states` (`moderngl_window.context.base.window.BaseWindow`  
`attribute`), 38
  - `mouse_states` (`moderngl_window.context.glfw.window.Window`  
`attribute`), 43
  - `mouse_states` (`moderngl_window.context.headless.window.Window`  
`attribute`), 50
  - `mouse_states` (`moderngl_window.context.pyglet.window.Window`  
`attribute`), 57
  - `mouse_states` (`moderngl_window.context.pyqt5.window.Window`  
`attribute`), 64
  - `mouse_states` (`moderngl_window.context.pyside2.window.Window`  
`attribute`), 71
  - `mouse_states` (`moderngl_window.context.sdl2.window.Window`  
`attribute`), 77
  - `mouse_wheel_event()` (`moderngl_window.context.pyqt5.window.Window`  
`method`), 60
  - `mouse_wheel_event()` (`moderngl_window.context.pyside2.window.Window`  
`method`), 66
  - `move_backward()` (`moderngl_window.scene.KeyboardCamera`  
`method`), 123
  - `move_down()` (`moderngl_window.scene.KeyboardCamera`  
`method`), 123
  - `move_forward()` (`moderngl_window.scene.KeyboardCamera`  
`method`), 123
  - `move_left()` (`moderngl_window.scene.KeyboardCamera`  
`method`), 123
  - `move_right()` (`moderngl_window.scene.KeyboardCamera`  
`method`), 123
  - `move_state()` (`moderngl_window.scene.KeyboardCamera`  
`method`), 124
  - `move_up()` (`moderngl_window.scene.KeyboardCamera`  
`method`), 123
- ## N
- `name` (`moderngl_window.scene.Material` `attribute`), 128
  - `near` (`moderngl_window.opengl.projection.Projection3D`  
`attribute`), 110
  - `next_frame()` (`moderngl_window.timers.base.BaseTimer` `method`), 119
  - `next_frame()` (`moderngl_window.timers.clock.Timer`  
`method`), 120
  - `Node` (in module `moderngl_window.scene`), 126
- ## O
- `on_close()` (`moderngl_window.context.pyglet.window.Window`  
`method`), 53
  - `on_hide()` (`moderngl_window.context.pyglet.window.Window`  
`method`), 53
  - `on_key_press()` (`moderngl_window.context.pyglet.window.Window`  
`method`), 52
  - `on_key_release()` (`moderngl_window.context.pyglet.window.Window`  
`method`), 52
  - `on_mouse_drag()` (`moderngl_window.context.pyglet.window.Window`  
`method`), 52
  - `on_mouse_motion()` (`moderngl_window.context.pyglet.window.Window`  
`method`), 53
  - `on_mouse_press()` (`moderngl_window.context.pyglet.window.Window`  
`method`), 52
  - `on_mouse_release()` (`moderngl_window.context.pyglet.window.Window`  
`method`), 53
  - `on_mouse_scroll()` (`moderngl_window.context.pyglet.window.Window`  
`method`), 53
  - `on_resize()` (`moderngl_window.context.pyglet.window.Window`  
`method`), 53
  - `on_show()` (`moderngl_window.context.pyglet.window.Window`  
`method`), 53
  - `on_text()` (`moderngl_window.context.pyglet.window.Window`  
`method`), 53
- ## P
- `parse_args()` (in module `moderngl_window`), 20
  - `path` (`moderngl_window.meta.base.ResourceDescription`  
`attribute`), 95
  - `path` (`moderngl_window.meta.data.DataDescription` `attribute`), 103

[path\(moderngl\\_window.meta.program.ProgramDescription attribute\), 100](#)  
[path\(moderngl\\_window.meta.scene.SceneDescription attribute\), 101](#)  
[path\(moderngl\\_window.meta.texture.TextureDescription attribute\), 97](#)  
[pause\(\)\(moderngl\\_window.timers.base.BaseTimer method\), 119](#)  
[pause\(\)\(moderngl\\_window.timers.clock.Timer method\), 120](#)  
[pixel\\_ratio\(moderngl\\_window.context.base.window.BaseWindow attribute\), 35](#)  
[pixel\\_ratio\(moderngl\\_window.context.glfw.window.Window attribute\), 40](#)  
[pixel\\_ratio\(moderngl\\_window.context.headless.window.Window attribute\), 48](#)  
[pixel\\_ratio\(moderngl\\_window.context.pyglet.window.Window attribute\), 55](#)  
[pixel\\_ratio\(moderngl\\_window.context.pyqt5.window.Window attribute\), 61](#)  
[pixel\\_ratio\(moderngl\\_window.context.pyside2.window.Window attribute\), 68](#)  
[pixel\\_ratio\(moderngl\\_window.context.sdl2.window.Window attribute\), 74](#)  
[position\(moderngl\\_window.context.base.window.BaseWindow attribute\), 34](#)  
[position\(moderngl\\_window.context.glfw.window.Window attribute\), 40](#)  
[position\(moderngl\\_window.context.headless.window.Window attribute\), 47](#)  
[position\(moderngl\\_window.context.pyglet.window.Window attribute\), 54](#)  
[position\(moderngl\\_window.context.pyqt5.window.Window attribute\), 61](#)  
[position\(moderngl\\_window.context.pyside2.window.Window attribute\), 68](#)  
[position\(moderngl\\_window.context.sdl2.window.Window attribute\), 74](#)  
[prepare\(\)\(moderngl\\_window.scene.Scene method\), 125](#)  
[print\\_context\\_info\(\)\(moderngl\\_window.context.base.window.BaseWindow method\), 34](#)  
[print\\_context\\_info\(\)\(moderngl\\_window.context.glfw.window.Window method\), 39](#)  
[print\\_context\\_info\(\)\(moderngl\\_window.context.headless.window.Window method\), 46](#)  
[print\\_context\\_info\(\)\(moderngl\\_window.context.pyglet.window.Window method\), 52](#)  
[print\\_context\\_info\(\)\(moderngl\\_window.context.pyqt5.window.Window method\), 59](#)  
[print\\_context\\_info\(\)\(moderngl\\_window.context.pyside2.window.Window method\), 66](#)  
[print\\_context\\_info\(\)\(moderngl\\_window.context.sdl2.window.Window method\), 73](#)  
[process\\_events\(\)\(moderngl\\_window.context.sdl2.window.Window method\), 73](#)  
[PROGRAM\\_DIRS\(moderngl\\_window.conf.Settings attribute\), 24](#)  
[PROGRAM\\_FINDERS\(moderngl\\_window.conf.Settings attribute\), 23](#)  
[PROGRAM\\_LOADERS\(moderngl\\_window.conf.Settings attribute\), 24](#)  
[ProgramDescription \(in module moderngl\\_window.meta.program\), 98](#)  
[Programs \(in module moderngl\\_window.resources.programs\), 115](#)  
[Projection \(moderngl\\_window.scene.Camera attribute\), 122](#)  
[Projection \(moderngl\\_window.scene.KeyboardCamera attribute\), 124](#)  
[Projection3D \(in module moderngl\\_window.opengl.projection\), 109](#)  
[projection\\_constants \(moderngl\\_window.opengl.projection.Projection3D attribute\), 110](#)

**Q**

[quad\\_2d\(\)\(in module moderngl\\_window.geometry\), 79](#)  
[quad\\_fs\(\)\(in module moderngl\\_window.geometry\), 79](#)

**R**

[register\\_data\\_dir\(\)\(in module moderngl\\_window.resources\), 113](#)  
[register\\_dir\(\)\(in module moderngl\\_window.resources\), 113](#)  
[register\\_program\\_dir\(\)\(in module moderngl\\_window.resources\), 113](#)  
[register\\_scene\\_dir\(\)\(in module moderngl\\_window.resources\), 113](#)  
[register\\_texture\\_dir\(\)\(in module moderngl\\_window.resources\), 113](#)  
[release\(\)\(moderngl\\_window.opengl.vao.VAO method\), 112](#)  
[reloadable\(moderngl\\_window.meta.program.ProgramDescription attribute\), 99](#)  
[render\(\)\(moderngl\\_window.context.base.window.BaseWindow method\), 33](#)



`render()` (`moderngl_window.context.base.window.WindowConfig` method), 39  
     method), 28  
`render()` (`moderngl_window.context.glfw.window.Window` method), 46  
     method), 39  
`render()` (`moderngl_window.context.headless.window.Window` method), 52  
     method), 46  
`render()` (`moderngl_window.context.pyglet.window.Window` method), 59  
     method), 52  
`render()` (`moderngl_window.context.pyqt5.window.Window` method), 66  
     method), 59  
`render()` (`moderngl_window.context.pyside2.window.Window` method), 72  
     method), 65  
`render()` (`moderngl_window.context.sdl2.window.Window` method), 72  
     method), 72  
`render()` (`moderngl_window.opengl.vao.VAO` method), 111  
`render_func` (`moderngl_window.context.base.window.BaseWindow` attribute), 49  
     attribute), 37  
`render_func` (`moderngl_window.context.glfw.window.Window` attribute), 57  
     attribute), 42  
`render_func` (`moderngl_window.context.headless.window.Window` attribute), 63  
     attribute), 49  
`render_func` (`moderngl_window.context.pyglet.window.Window` attribute), 70  
     attribute), 57  
`render_func` (`moderngl_window.context.pyqt5.window.Window` attribute), 76  
     attribute), 63  
`render_func` (`moderngl_window.context.pyside2.window.Window` attribute), 70  
     attribute), 70  
`render_func` (`moderngl_window.context.sdl2.window.Window` attribute), 76  
     attribute), 76  
`render_indirect()` (`moderngl_window.opengl.vao.VAO` method), 111  
     111  
`resizable` (`moderngl_window.context.base.window.BaseWindow` method), 115  
     attribute), 35  
`resizable` (`moderngl_window.context.base.window.WindowConfig` method), 116  
     attribute), 31  
`resizable` (`moderngl_window.context.glfw.window.Window` method), 115  
     attribute), 41  
`resizable` (`moderngl_window.context.headless.window.Window` method), 115  
     attribute), 48  
`resizable` (`moderngl_window.context.pyglet.window.Window` method), 115  
     attribute), 55  
`resizable` (`moderngl_window.context.pyqt5.window.Window` method), 103  
     attribute), 62  
`resizable` (`moderngl_window.context.pyside2.window.Window` method), 103  
     attribute), 69  
`resizable` (`moderngl_window.context.sdl2.window.Window` method), 100  
     attribute), 74  
`resize()` (`moderngl_window.context.base.window.BaseWindow` method), 33  
     method), 33  
`resize()` (`moderngl_window.context.base.window.WindowConfig` method), 101  
     method), 28  
`resize()` (`moderngl_window.context.glfw.window.Window` method), 46  
     method), 39  
`resize()` (`moderngl_window.context.pyglet.window.Window` method), 59  
     method), 52  
`resize()` (`moderngl_window.context.pyqt5.window.Window` method), 66  
     method), 59  
`resize()` (`moderngl_window.context.pyside2.window.Window` method), 72  
     method), 65  
`resize_func` (`moderngl_window.context.base.window.BaseWindow` attribute), 37  
     attribute), 37  
`resize_func` (`moderngl_window.context.glfw.window.Window` attribute), 57  
     attribute), 42  
`resize_func` (`moderngl_window.context.pyglet.window.Window` attribute), 70  
     attribute), 57  
`resize_func` (`moderngl_window.context.pyside2.window.Window` attribute), 70  
     attribute), 70  
`resize_func` (`moderngl_window.context.sdl2.window.Window` attribute), 76  
     attribute), 76  
`resolve_loader()` (`moderngl_window.resources.base.BaseRegistry` method), 114  
     method), 114  
`resolve_loader()` (`moderngl_window.resources.data.DataFiles` method), 117  
     method), 117  
`resolve_loader()` (`moderngl_window.resources.programs.Programs` method), 117  
     method), 117  
`resolve_loader()` (`moderngl_window.resources.scenes.Scenes` method), 116  
     method), 116  
`resolve_loader()` (`moderngl_window.resources.textures.Textures` method), 115  
     method), 115  
`resolved_path` (`moderngl_window.meta.base.ResourceDescription` attribute), 95  
     attribute), 95  
`resolved_path` (`moderngl_window.meta.data.DataDescription` attribute), 103  
     attribute), 103  
`resolved_path` (`moderngl_window.meta.program.ProgramDescription` attribute), 100  
     attribute), 100  
`resolved_path` (`moderngl_window.meta.scene.SceneDescription` attribute), 101  
     attribute), 101  
`resolved_path` (`moderngl_window.meta.texture.TextureDescription` attribute), 101  
     attribute), 101

[attribute](#)), 97  
[resource\\_dir](#) (mod-[erngl\\_window.resources.scenes](#)), 116  
[erngl\\_window.context.base.window.WindowConfig](#)[SCREENSHOT\\_PATH](#) (mod-[erngl\\_window.conf.Settings](#)[attribute](#)), 23  
[resource\\_type](#) (mod-[erngl\\_window.meta.base.ResourceDescription](#)[attribute](#)), 96  
[resource\\_type](#) (mod-[erngl\\_window.meta.data.DataDescription](#)[attribute](#)), 103  
[resource\\_type](#) (mod-[erngl\\_window.meta.program.ProgramDescription](#)[attribute](#)), 100  
[resource\\_type](#) (mod-[erngl\\_window.meta.scene.SceneDescription](#)[attribute](#)), 102  
[resource\\_type](#) (mod-[erngl\\_window.meta.texture.TextureDescription](#)[attribute](#)), 98  
[ResourceDescription](#) (in module mod-[erngl\\_window.meta.base](#)), 95  
[rot\\_state\(\)](#) (mod-[erngl\\_window.scene.KeyboardCamera](#)[method](#)), 124  
[run\\_window\\_config\(\)](#) (in module mod-[erngl\\_window](#)), 20

## S

[sampler](#) (mod-[erngl\\_window.scene.MaterialTexture](#)[attribute](#)), 129  
[samples](#) (mod-[erngl\\_window.context.base.window.BaseWindow](#)[attribute](#)), 36  
[samples](#) (mod-[erngl\\_window.context.base.window.WindowConfig](#)[attribute](#)), 32  
[samples](#) (mod-[erngl\\_window.context.glfw.window.Window](#)[attribute](#)), 42  
[samples](#) (mod-[erngl\\_window.context.headless.window.Window](#)[attribute](#)), 49  
[samples](#) (mod-[erngl\\_window.context.pyglet.window.Window](#)[attribute](#)), 56  
[samples](#) (mod-[erngl\\_window.context.pyqt5.window.Window](#)[attribute](#)), 63  
[samples](#) (mod-[erngl\\_window.context.pyside2.window.Window](#)[attribute](#)), 69  
[samples](#) (mod-[erngl\\_window.context.sdl2.window.Window](#)[attribute](#)), 75  
[SCENE\\_DIRS](#) (mod-[erngl\\_window.conf.Settings](#)[attribute](#)), 24  
[SCENE\\_FINDERS](#) (mod-[erngl\\_window.conf.Settings](#)[attribute](#)), 24  
[SCENE\\_LOADERS](#) (mod-[erngl\\_window.conf.Settings](#)[attribute](#)), 24  
[SceneDescription](#) (in module mod-[erngl\\_window.meta.scene](#)), 100

[Scenes](#) (in module mod-[erngl\\_window.resources.scenes](#)), 116  
[set\\_default\\_viewport\(\)](#) (mod-[erngl\\_window.context.base.window.BaseWindow](#)[method](#)), 33  
[set\\_default\\_viewport\(\)](#) (mod-[erngl\\_window.context.glfw.window.Window](#)[method](#)), 39  
[set\\_default\\_viewport\(\)](#) (mod-[erngl\\_window.context.headless.window.Window](#)[method](#)), 46  
[set\\_default\\_viewport\(\)](#) (mod-[erngl\\_window.context.pyglet.window.Window](#)[method](#)), 52  
[set\\_default\\_viewport\(\)](#) (mod-[erngl\\_window.context.pyqt5.window.Window](#)[method](#)), 59  
[set\\_default\\_viewport\(\)](#) (mod-[erngl\\_window.context.pyside2.window.Window](#)[method](#)), 66  
[set\\_default\\_viewport\(\)](#) (mod-[erngl\\_window.context.sdl2.window.Window](#)[method](#)), 72  
[set\\_position\(\)](#) (mod-[erngl\\_window.scene.Camera](#)[method](#)), 121  
[set\\_position\(\)](#) (mod-[erngl\\_window.scene.KeyboardCamera](#)[method](#)), 123  
[Settings](#) (in module mod-[erngl\\_window.conf](#)), 21  
[Settings\\_attr](#) (mod-[erngl\\_window.finders.base.BaseFilesystemFinder](#)[attribute](#)), 105  
[settings\\_attr](#) (mod-[erngl\\_window.finders.data.FilesystemFinder](#)[attribute](#)), 107  
[settings\\_attr](#) (mod-[erngl\\_window.finders.program.FilesystemFinder](#)[attribute](#)), 106  
[settings\\_attr](#) (mod-[erngl\\_window.finders.scene.FilesystemFinder](#)[attribute](#)), 106  
[settings\\_attr](#) (mod-[erngl\\_window.finders.texture.FilesystemFinder](#)[attribute](#)), 106  
[settings\\_attr](#) (mod-[erngl\\_window.resources.base.BaseRegistry](#)[attribute](#)), 114  
[settings\\_attr](#) (mod-[erngl\\_window.resources.data.DataFiles](#)[attribute](#)), 117  
[settings\\_attr](#) (mod-[erngl\\_window.resources.programs.Programs](#)[attribute](#)), 117

attribute), 116  
 settings\_attr (moderngl\_window.resources.scenes.Scenes attribute), 117  
 settings\_attr (moderngl\_window.resources.textures.Textures attribute), 115  
 setup\_basic\_logging() (in module moderngl\_window), 19  
 show\_event() (moderngl\_window.context.pyqt5.window.Window method), 60  
 show\_event() (moderngl\_window.context.pyside2.window.Window method), 67  
 size(moderngl\_window.context.base.window.BaseWindow attribute), 34  
 size(moderngl\_window.context.glfw.window.Window attribute), 40  
 size(moderngl\_window.context.headless.window.Window attribute), 47  
 size(moderngl\_window.context.pyglet.window.Window attribute), 54  
 size(moderngl\_window.context.pyqt5.window.Window attribute), 61  
 size(moderngl\_window.context.pyside2.window.Window attribute), 67  
 size(moderngl\_window.context.sdl2.window.Window attribute), 73  
 sphere() (in module moderngl\_window.geometry), 80  
 start() (moderngl\_window.timers.base.BaseTimer method), 119  
 start() (moderngl\_window.timers.clock.Timer method), 120  
 stop() (moderngl\_window.timers.base.BaseTimer method), 119  
 stop() (moderngl\_window.timers.clock.Timer method), 120  
 supported\_extensions (moderngl\_window.loaders.scene.gltf2.Loader attribute), 89  
 supports\_file() (moderngl\_window.loaders.base.BaseLoader class method), 81  
 supports\_file() (moderngl\_window.loaders.data.binary.Loader class method), 93  
 supports\_file() (moderngl\_window.loaders.data.json.Loader class method), 91  
 supports\_file() (moderngl\_window.loaders.data.text.Loader class method), 92  
 supports\_file() (moderngl\_window.loaders.program.separate.Loader class method), 85  
 supports\_file() (moderngl\_window.loaders.program.single.Loader class method), 83  
 supports\_file() (moderngl\_window.loaders.scene.gltf2.Loader class method), 88  
 supports\_file() (moderngl\_window.loaders.scene.stl.Loader class method), 90  
 supports\_file() (moderngl\_window.loaders.scene.wavefront.Loader class method), 87  
 supports\_file() (moderngl\_window.loaders.texture.array.Loader class method), 86  
 supports\_file() (moderngl\_window.loaders.texture.t2d.Loader class method), 82  
 swap\_buffers() (moderngl\_window.context.base.window.BaseWindow method), 33  
 swap\_buffers() (moderngl\_window.context.glfw.window.Window method), 39  
 swap\_buffers() (moderngl\_window.context.headless.window.Window method), 46  
 swap\_buffers() (moderngl\_window.context.pyglet.window.Window method), 52  
 swap\_buffers() (moderngl\_window.context.pyqt5.window.Window method), 59  
 swap\_buffers() (moderngl\_window.context.pyside2.window.Window method), 66  
 swap\_buffers() (moderngl\_window.context.sdl2.window.Window method), 72

**T**

tess\_control\_shader (moderngl\_window.meta.program.ProgramDescription attribute), 99  
 tess\_evaluation\_shader (moderngl\_window.meta.program.ProgramDescription attribute), 99  
 texture(moderngl\_window.scene.MaterialTexture attribute), 129  
 TEXTURE\_DIRS(moderngl\_window.conf.Settings attribute), 24



TEXTURE\_FINDERS (*moderngl\_window.conf.Settings attribute*), 24

TEXTURE\_LOADERS (*moderngl\_window.conf.Settings attribute*), 24

TextureDescription (*in module moderngl\_window.meta.texture*), 96

Textures (*in module moderngl\_window.resources.textures*), 114

time (*moderngl\_window.timers.base.BaseTimer attribute*), 120

time (*moderngl\_window.timers.clock.Timer attribute*), 120

Timer (*in module moderngl\_window.timers.clock*), 120

title (*moderngl\_window.context.base.window.BaseWindow attribute*), 34

title (*moderngl\_window.context.base.window.WindowConfig attribute*), 31

title (*moderngl\_window.context.glfw.window.Window attribute*), 40

title (*moderngl\_window.context.headless.window.Window attribute*), 47

title (*moderngl\_window.context.pyglet.window.Window attribute*), 54

title (*moderngl\_window.context.pyqt5.window.Window attribute*), 60

title (*moderngl\_window.context.pyside2.window.Window attribute*), 67

title (*moderngl\_window.context.sdl2.window.Window attribute*), 73

to\_dict() (*moderngl\_window.conf.Settings method*), 22

tobytes() (*moderngl\_window.opengl.projection.Projection3D method*), 109

toggle\_pause() (*moderngl\_window.timers.base.BaseTimer method*), 119

toggle\_pause() (*moderngl\_window.timers.clock.Timer method*), 120

transform() (*moderngl\_window.opengl.vao.VAO method*), 111

**U**

unicode\_char\_entered() (*moderngl\_window.context.base.window.WindowConfig method*), 29

unicode\_char\_entered\_func (*moderngl\_window.context.base.window.BaseWindow attribute*), 37

unicode\_char\_entered\_func (*moderngl\_window.context.glfw.window.Window attribute*), 43

unicode\_char\_entered\_func (*moderngl\_window.context.headless.window.Window attribute*), 50

unicode\_char\_entered\_func (*moderngl\_window.context.pyglet.window.Window attribute*), 57

unicode\_char\_entered\_func (*moderngl\_window.context.pyqt5.window.Window attribute*), 64

unicode\_char\_entered\_func (*moderngl\_window.context.pyside2.window.Window attribute*), 71

unicode\_char\_entered\_func (*moderngl\_window.context.sdl2.window.Window attribute*), 76

update() (*moderngl\_window.opengl.projection.Projection3D method*), 109

use() (*moderngl\_window.context.base.window.BaseWindow method*), 33

use() (*moderngl\_window.context.glfw.window.Window method*), 39

use() (*moderngl\_window.context.headless.window.Window method*), 46

use() (*moderngl\_window.context.pyglet.window.Window method*), 51

use() (*moderngl\_window.context.pyqt5.window.Window method*), 58

use() (*moderngl\_window.context.pyside2.window.Window method*), 65

use() (*moderngl\_window.context.sdl2.window.Window method*), 72

**V**

VAO (*in module moderngl\_window.opengl.vao*), 110

velocity (*moderngl\_window.scene.KeyboardCamera attribute*), 124

vertex\_shader (*moderngl\_window.meta.program.ProgramDescription attribute*), 99

viewport (*moderngl\_window.context.base.window.BaseWindow attribute*), 35

viewport (*moderngl\_window.context.glfw.window.Window attribute*), 41

viewport (*moderngl\_window.context.headless.window.Window attribute*), 48

viewport (*moderngl\_window.context.pyglet.window.Window attribute*), 55

viewport (*moderngl\_window.context.pyqt5.window.Window attribute*), 61

viewport (*moderngl\_window.context.pyside2.window.Window attribute*), 68

viewport (*moderngl\_window.context.sdl2.window.Window attribute*), 74

viewport\_height (*moderngl\_window.context.base.window.BaseWindow attribute*), 35

viewport_height (moderngl_window.context.glfw.window.Window attribute), 41	viewport_width (moderngl_window.context.pyside2.window.Window attribute), 68
viewport_height (moderngl_window.context.headless.window.Window attribute), 48	viewport_width (moderngl_window.context.sdl2.window.Window attribute), 74
viewport_height (moderngl_window.context.pyglet.window.Window attribute), 55	vsync (moderngl_window.context.base.window.BaseWindow attribute), 36
viewport_height (moderngl_window.context.pyqt5.window.Window attribute), 62	vsync (moderngl_window.context.glfw.window.Window attribute), 41
viewport_height (moderngl_window.context.pyside2.window.Window attribute), 68	vsync (moderngl_window.context.headless.window.Window attribute), 48
viewport_height (moderngl_window.context.sdl2.window.Window attribute), 74	vsync (moderngl_window.context.pyglet.window.Window attribute), 56
viewport_size (moderngl_window.context.base.window.BaseWindow attribute), 35	vsync (moderngl_window.context.pyqt5.window.Window attribute), 62
viewport_size (moderngl_window.context.glfw.window.Window attribute), 41	vsync (moderngl_window.context.pyside2.window.Window attribute), 69
viewport_size (moderngl_window.context.headless.window.Window attribute), 48	vsync (moderngl_window.context.sdl2.window.Window attribute), 75
viewport_size (moderngl_window.context.pyglet.window.Window attribute), 55	
viewport_size (moderngl_window.context.pyqt5.window.Window attribute), 61	
viewport_size (moderngl_window.context.pyside2.window.Window attribute), 68	
viewport_size (moderngl_window.context.sdl2.window.Window attribute), 74	
viewport_width (moderngl_window.context.base.window.BaseWindow attribute), 35	
viewport_width (moderngl_window.context.glfw.window.Window attribute), 41	
viewport_width (moderngl_window.context.headless.window.Window attribute), 48	
viewport_width (moderngl_window.context.pyglet.window.Window attribute), 55	
viewport_width (moderngl_window.context.pyqt5.window.Window attribute), 62	

**W**

width (moderngl_window.context.base.window.BaseWindow attribute), 34
width (moderngl_window.context.glfw.window.Window attribute), 40
width (moderngl_window.context.headless.window.Window attribute), 47
width (moderngl_window.context.pyglet.window.Window attribute), 54
width (moderngl_window.context.pyqt5.window.Window attribute), 61
width (moderngl_window.context.pyside2.window.Window attribute), 67
width (moderngl_window.context.sdl2.window.Window attribute), 73
WINDOW (moderngl_window.conf.Settings attribute), 23
window() (in module moderngl_window), 19
window_size (moderngl_window.context.base.window.WindowConfig attribute), 31
WindowConfig (in module moderngl_window.context.base.window), 27