
moderngl_window Documentation

Release 1.4.0

Einar Forseiv

Nov 11, 2019

PROGRAMMING GUIDE

1	Installation	3
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
2	Basic Usage (WindowConfig)	5
2.1	Basic Example	5
2.2	Command Line Arguments	6
2.3	Resource Loading	6
2.4	Generic events and window types	6
2.5	Window Events	6
2.6	Keyboard Input	7
2.7	Mouse Input	7
3	Custom Usage	9
3.1	Register the moderngl.Context	9
3.2	Register Resource Directories	9
3.3	Using Built In Window Types	10
4	The Resource System	11
4.1	Resource Types	11
4.2	Resource Paths	11
4.3	Resource Descriptions	12
4.4	Loading Resources	12
4.4.1	Textures	13
4.4.2	Programs	13
4.4.3	Scenes	13
4.4.4	Data	13
5	moderngl_window	15
6	moderngl_window.conf.Settings	17
6.1	Methods	17
6.2	Attributes	19
7	moderngl_window.context	23
7.1	base.window.WindowConfig	23
7.1.1	Methods	24
7.1.2	Attributes	27

7.2	base.BaseWindow	28
7.2.1	Methods	28
7.2.2	Attributes	29
7.3	glfw.Glfw	32
7.3.1	Methods	32
7.3.2	Window Specific Methods	34
7.3.3	Attributes	35
7.4	headless.Headless	38
7.4.1	Methods	38
7.4.2	Attributes	39
7.5	pyglet.Window	42
7.5.1	Methods	42
7.5.2	Window Specific Methods	43
7.5.3	Attributes	44
7.6	pyqt5Window	47
7.6.1	Methods	47
7.6.2	Window Specific Methods	49
7.6.3	Attributes	49
7.7	pyside2Window	52
7.7.1	Methods	52
7.7.2	Window Specific Methods	54
7.7.3	Attributes	54
7.8	sdl2.Window	57
7.8.1	Methods	57
7.8.2	Window Specific Methods	59
7.8.3	Attributes	59
8	moderngl_window.geometry	63
9	moderngl_window.loaders	65
9.1	base.BaseLoader	65
9.1.1	Method	65
9.1.2	Attributes	66
9.2	texture.t2d.Loader	66
9.2.1	Method	66
9.2.2	Attributes	67
9.3	program.single.Loader	67
9.3.1	Method	67
9.3.2	Attributes	68
9.4	program.separate.Loader	69
9.4.1	Method	69
9.4.2	Loader Specific Methods	70
9.4.3	Attributes	70
9.5	texture.array.Loader	70
9.5.1	Method	70
9.5.2	Attributes	71
9.6	scene.wavefront.Loader	71
9.6.1	Method	71
9.6.2	Attributes	72
9.7	scene.glTF2.Loader	72
9.7.1	Method	72
9.7.2	Loader Specific Methods	73
9.7.3	Attributes	73
9.7.4	Loader Specific Attributes	73

9.8	scene.stl.Loader	74
9.8.1	Method	74
9.8.2	Attributes	74
9.9	data.json.Loader	75
9.9.1	Method	75
9.9.2	Attributes	75
9.10	data.text.Loader	76
9.10.1	Method	76
9.10.2	Attributes	76
9.11	data.binary.Loader	77
9.11.1	Method	77
9.11.2	Attributes	77
10	moderngl_window.meta	79
10.1	base.ResourceDescription	79
10.1.1	Methods	79
10.1.2	Attributes	79
10.2	texture.TextureDescription	80
10.2.1	Methods	80
10.2.2	Attributes	81
10.2.3	Inherited Attributes	81
10.3	program.ProgramDescription	82
10.3.1	Methods	83
10.3.2	Attributes	83
10.3.3	Inherited Attributes	84
10.4	scene.SceneDescription	84
10.4.1	Methods	85
10.4.2	Attributes	85
10.4.3	Inherited Attributes	85
10.5	data.DataDescription	86
10.5.1	Methods	87
10.5.2	Attributes	87
11	moderngl_window.finders	89
11.1	base.BaseFilesystemFinder	89
11.1.1	Methods	89
11.1.2	Attributes	89
11.2	texture.FilesystemFinder	89
11.2.1	Methods	89
11.2.2	Attributes	90
11.3	program.FilesystemFinder	90
11.3.1	Methods	90
11.3.2	Attributes	90
11.4	scene.FilesystemFinder	90
11.4.1	Methods	90
11.4.2	Attributes	90
11.5	data.FilesystemFinder	91
11.5.1	Methods	91
11.5.2	Attributes	91
12	moderngl_window.opengl	93
12.1	opengl.projection.Projection3D	93
12.1.1	Methods	93
12.1.2	Attributes	93

12.2	opengl.vao.VAO	94
12.2.1	Methods	95
12.2.2	Attributes	96
13	moderngl_window.resources	97
13.1	base.BaseRegistry	97
13.1.1	Methods	97
13.1.2	Attributes	98
13.2	textures.Textures	98
13.2.1	Methods	98
13.2.2	Attributes	99
13.3	programs.Programs	99
13.3.1	Methods	99
13.3.2	Attributes	100
13.4	scenes.Scenes	100
13.4.1	Methods	100
13.4.2	Attributes	101
13.5	base.DataFiles	101
13.5.1	Methods	101
13.5.2	Attributes	101
14	moderngl_window.timers	103
14.1	base.BaseTimer	103
14.1.1	Methods	103
14.1.2	Attributes	103
14.2	clock.Timer	104
14.2.1	Methods	104
14.2.2	Attributes	104
15	moderngl_window.scene	107
15.1	Camera	107
15.1.1	Methods	107
15.1.2	Attributes	108
15.2	KeyboardCamera	108
15.2.1	Methods	108
15.2.2	Inherited Methods	109
15.2.3	Attributes	109
15.3	Scene	110
15.3.1	Methods	110
15.3.2	Attributes	110
15.4	Node	111
15.4.1	Methods	111
15.4.2	Attributes	112
15.5	Mesh	112
15.5.1	Methods	112
15.6	Material	113
15.6.1	Methods	113
15.6.2	Attributes	113
15.7	MaterialTexture	114
15.7.1	Methods	114
15.7.2	Attributes	114
15.8	MeshProgram	114
15.8.1	Methods	114
15.8.2	Attributes	115

16 Indices and tables	117
Python Module Index	119
Index	121

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
# Package name with dash also works
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 avaialble 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 enviroment
tox -e py35
tox -e py36
tox -e py37

# pep8 run
tox -e pep8

# Run all enviroments
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 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.

2.3 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.

2.4 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 suboptimally 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.

2.5 Window Events

Implement the `resize` method to customize window resize handling.

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

2.6 Keyboard Input

Implement the `key_event` 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")
```

2.7 Mouse Input

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

```
def mouse_position_event(self, x, y):  
    print("Mouse position:", x, y)  
  
def mouse_drag_event(self, x, y):  
    print("Mouse drag:", 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))
```


CUSTOM USAGE

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

3.1 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.

3.2 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.

3.3 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),
    ...,
)
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.

THE RESOURCE SYSTEM

4.1 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.

4.2 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.

4.3 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.

4.4 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,
)
```

4.4.1 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))
```

4.4.2 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',
    )
)
```

4.4.3 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"))
```

4.4.4 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.

Keyword Arguments

- **window** (*window*) – The currently active window
- **ctx** (*moderngl.Context*) – The active moderngl context

`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)
```

6.1 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 minimnal 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 enviroment 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

6.2 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

7.1 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:

```
class MyConfig(mglw.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):
        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)
```

7.1.1 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.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)

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

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 in 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*)

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

Parameters

- **x**(*int*) – X position of the mouse cursor
- **Iint**(*y*) – Y position of the mouse cursor

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
- **Iint**(*y_offset*) – 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

7.1.2 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 endorsed 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
```

7.2 base.BaseWindow

7.2.1 Methods

`BaseWindow.__init__(title='ModernGL', gl_version=(3, 3), size=(1280, 720), resizable=True, fullscreen=False, vsync=True, aspect_ratio=1.7777777777777777, 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 aspect ratio. Can be set to `None`.
- **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 viewport based on the configured aspect ratio. Will add black borders and center the viewport if the window do not match the configured viewport.

If aspect ratio is None the viewport will be scaled to the entire window size regardless of size.

`BaseWindow.print_context_info()`

Prints moderngl context info.

7.2.2 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

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

Type `Tuple[int, 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.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

`BaseWindow.vsync`

vertical sync enabled/disabled

Type bool

`BaseWindow.aspect_ratio`

Aspect ratio configured for the viewport

Type float

`BaseWindow.samples`

Number of Multisample anti-aliasing (MSAA) samples

Type float

`BaseWindow.cursor`

Should the mouse cursor be visible inside the window?

Type bool

`BaseWindow.render_func`

The render callable

This property can also be used to assign a callable.

Type callable

`BaseWindow.resize_func`

The resize callable

This property can also be used to assign a callable.

Type callable

`BaseWindow.key_event_func`

The key_event callable

This property can also be used to assign a callable.

Type callable

`BaseWindow.mouse_position_event_func`

The mouse_position callable

This property can also be used to assign a callable.

Type callable

`BaseWindow.mouse_press_event_func`

The mouse_press callable

This property can also be used to assign a callable.

Type callable

`BaseWindow.mouse_release_event_func`

The mouse_release callable

This property can also be used to assign a callable.

Type callable

`BaseWindow.mouse_drag_event_func`

The mouse_drag callable

This property can also be used to assign a callable.

Type callable

`BaseWindow.mouse_scroll_event_func`

The `mouse_scroll_event` callable

This property can also be used to assign a callable.

Type callable

`BaseWindow.unicode_char_entered_func`

The `unicode_char_entered` callable

This property can also be used to assign a 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

7.3 glfw.GLFW

7.3.1 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 aspect ratio. Can be set to None.
- **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 viewport based on the configured aspect ratio. Will add black borders and center the viewport if the window do not match the configured viewport.

If aspect ratio is None the viewport will be scaled to the entire window size regardless of size.

Window.**print_context_info**()

Prints moderngl context info.

7.3.2 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

7.3.3 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

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

Type Tuple[int, 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.**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

Window.**vsync**

vertical sync enabled/disabled

Type bool

Window.**aspect_ratio**

Aspect ratio configured for the viewport

Type float

Window.**samples**

Number of Multisample anti-aliasing (MSAA) samples

Type float

Window.**cursor**

Should the mouse cursor be visible inside the window?

Type bool

Window.**render_func**

The render callable

This property can also be used to assign a callable.

Type callable

Window.**resize_func**

The resize callable

This property can also be used to assign a callable.

Type callable

Window.**key_event_func**

The key_event callable

This property can also be used to assign a callable.

Type callable

Window.**mouse_position_event_func**

The mouse_position callable

This property can also be used to assign a callable.

Type callable

Window.**mouse_press_event_func**

The mouse_press callable

This property can also be used to assign a callable.

Type callable

`Window.mouse_release_event_func`

The mouse_release callable

This property can also be used to assign a callable.

Type callable

`Window.mouse_drag_event_func`

The mouse_drag callable

This property can also be used to assign a callable.

Type callable

`Window.mouse_scroll_event_func`

The mouse_scroll_event callable

This property can also be used to assign a callable.

Type callable

`Window.unicode_char_entered_func`

The unicode_char_entered callable

This property can also be used to assign a 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

7.4 headless.Headless

7.4.1 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 aspect ratio. Can be set to None.
- **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 viewport based on the configured aspect ratio. Will add black borders and center the viewport if the window do not match the configured viewport.

If aspect ratio is None the viewport will be scaled to the entire window size regardless of size.

Window.**print_context_info**()

Prints moderngl context info.

7.4.2 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

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

Type Tuple[int, 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.**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

Window.**vsync**

vertical sync enabled/disabled

Type bool

Window.**aspect_ratio**

Aspect ratio configured for the viewport

Type float

Window.**samples**

Number of Multisample anti-aliasing (MSAA) samples

Type float

Window.**cursor**

Should the mouse cursor be visible inside the window?

Type bool

Window.**render_func**

The render callable

This property can also be used to assign a callable.

Type callable

Window.**resize_func**

The resize callable

This property can also be used to assign a callable.

Type callable

Window.**key_event_func**

The key_event callable

This property can also be used to assign a callable.

Type callable

Window.**mouse_position_event_func**

The mouse_position callable

This property can also be used to assign a callable.

Type callable

Window.**mouse_press_event_func**

The mouse_press callable

This property can also be used to assign a callable.

Type callable

Window.**mouse_release_event_func**

The mouse_release callable

This property can also be used to assign a callable.

Type callable

Window.**mouse_drag_event_func**

The mouse_drag callable

This property can also be used to assign a callable.

Type callable

Window.**mouse_scroll_event_func**

The mouse_scroll_event callable

This property can also be used to assign a callable.

Type callable

Window.**unicode_char_entered_func**

The unicode_char_entered callable

This property can also be used to assign a 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`

7.5 pyglet.Window

7.5.1 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 aspect ratio. Can be set to `None`.
- **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 viewport based on the configured aspect ratio. Will add black borders and center the viewport if the window do not match the configured viewport.

If aspect ratio is None the viewport will be scaled to the entire window size regardless of size.

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

7.5.2 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

7.5.3 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
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
Type Tuple[int, 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.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

Window.vsync
vertical sync enabled/disabled
Type bool

Window.**aspect_ratio**

Aspect ratio configured for the viewport

Type float

Window.**samples**

Number of Multisample anti-aliasing (MSAA) samples

Type float

Window.**cursor**

Should the mouse cursor be visible inside the window?

Type bool

Window.**render_func**

The render callable

This property can also be used to assign a callable.

Type callable

Window.**resize_func**

The resize callable

This property can also be used to assign a callable.

Type callable

Window.**key_event_func**

The key_event callable

This property can also be used to assign a callable.

Type callable

Window.**mouse_position_event_func**

The mouse_position callable

This property can also be used to assign a callable.

Type callable

Window.**mouse_press_event_func**

The mouse_press callable

This property can also be used to assign a callable.

Type callable

Window.**mouse_release_event_func**

The mouse_release callable

This property can also be used to assign a callable.

Type callable

Window.**mouse_drag_event_func**

The mouse_drag callable

This property can also be used to assign a callable.

Type callable

Window.**unicode_char_entered_func**

The unicode_char_entered callable

This property can also be used to assign a callable.

Type callable

Window.**mouse_scroll_event_func**

The mouse_scroll_event calable

This property can also be used to assign a 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

7.6 pyqt5Window

7.6.1 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 aspect ratio. Can be set to None.
- **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

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

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 viewport based on the configured aspect ratio. Will add black borders and center the viewport if the window do not match the configured viewport.

If aspect ratio is None the viewport will be scaled to the entire window size regardless of size.

`Window.print_context_info()`

Prints moderngl context info.

7.6.2 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

7.6.3 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

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

Type Tuple[int, 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.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

Window.vsync

vertical sync enabled/disabled

Type bool

Window.aspect_ratio

Aspect ratio configured for the viewport

Type float

Window.**samples**

Number of Multisample anti-aliasing (MSAA) samples

Type float

Window.**cursor**

Should the mouse cursor be visible inside the window?

Type bool

Window.**render_func**

The render callable

This property can also be used to assign a callable.

Type callable

Window.**resize_func**

The resize callable

This property can also be used to assign a callable.

Type callable

Window.**key_event_func**

The key_event callable

This property can also be used to assign a callable.

Type callable

Window.**mouse_position_event_func**

The mouse_position callable

This property can also be used to assign a callable.

Type callable

Window.**mouse_press_event_func**

The mouse_press callable

This property can also be used to assign a callable.

Type callable

Window.**mouse_release_event_func**

The mouse_release callable

This property can also be used to assign a callable.

Type callable

Window.**mouse_drag_event_func**

The mouse_drag callable

This property can also be used to assign a callable.

Type callable

Window.**unicode_char_entered_func**

The unicode_char_entered callable

This property can also be used to assign a callable.

Type callable

Window.**mouse_scroll_event_func**

The mouse_scroll_event callable

This property can also be used to assign a 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

7.7 pyside2Window

7.7.1 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 aspect ratio. Can be set to None.
- **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

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

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 viewport based on the configured aspect ratio. Will add black borders and center the viewport if the window do not match the configured viewport.

If aspect ratio is None the viewport will be scaled to the entire window size regardless of size.

`Window.print_context_info()`

Prints moderngl context info.

7.7.2 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

7.7.3 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

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

Type Tuple[int, 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.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

Window.vsync

vertical sync enabled/disabled

Type bool

Window.aspect_ratio

Aspect ratio configured for the viewport

Type float

Window.**samples**

Number of Multisample anti-aliasing (MSAA) samples

Type float

Window.**cursor**

Should the mouse cursor be visible inside the window?

Type bool

Window.**render_func**

The render callable

This property can also be used to assign a callable.

Type callable

Window.**resize_func**

The resize callable

This property can also be used to assign a callable.

Type callable

Window.**key_event_func**

The key_event callable

This property can also be used to assign a callable.

Type callable

Window.**mouse_position_event_func**

The mouse_position callable

This property can also be used to assign a callable.

Type callable

Window.**mouse_press_event_func**

The mouse_press callable

This property can also be used to assign a callable.

Type callable

Window.**mouse_release_event_func**

The mouse_release callable

This property can also be used to assign a callable.

Type callable

Window.**mouse_drag_event_func**

The mouse_drag callable

This property can also be used to assign a callable.

Type callable

Window.**unicode_char_entered_func**

The unicode_char_entered callable

This property can also be used to assign a callable.

Type callable

Window.**mouse_scroll_event_func**

The mouse_scroll_event callable

This property can also be used to assign a 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

7.8 sdl2.Window

7.8.1 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 aspect ratio. Can be set to None.
- **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

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

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 viewport based on the configured aspect ratio. Will add black borders and center the viewport if the window do not match the configured viewport.

If aspect ratio is None the viewport will be scaled to the entire window size regardless of size.

`Window.print_context_info()`

Prints moderngl context info.

7.8.2 Window Specific Methods

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

7.8.3 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
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
Type `Tuple[int, 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.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

Window.**vsync**

vertical sync enabled/disabled

Type bool

Window.**aspect_ratio**

Aspect ratio configured for the viewport

Type float

Window.**samples**

Number of Multisample anti-aliasing (MSAA) samples

Type float

Window.**cursor**

Should the mouse cursor be visible inside the window?

Type bool

Window.**render_func**

The render callable

This property can also be used to assign a callable.

Type callable

Window.**resize_func**

The resize callable

This property can also be used to assign a callable.

Type callable

Window.**key_event_func**

The key_event callable

This property can also be used to assign a callable.

Type callable

Window.**mouse_position_event_func**

The mouse_position callable

This property can also be used to assign a callable.

Type callable

Window.**mouse_press_event_func**

The mouse_press callable

This property can also be used to assign a callable.

Type callable

`Window.mouse_release_event_func`

The mouse_release callable

This property can also be used to assign a callable.

Type callable

`Window.mouse_drag_event_func`

The mouse_drag callable

This property can also be used to assign a callable.

Type callable

`Window.unicode_char_entered_func`

The unicode_char_entered callable

This property can also be used to assign a callable.

Type callable

`Window.mouse_scroll_event_func`

The mouse_scroll_event callable

This property can also be used to assign a 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 coordiantes.

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 `demosys.opengl.vao.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 'moderngl_window.geometry.attributes.AttributeNames'>) → moderngl_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 'moderngl_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

9.1 base.BaseLoader

9.1.1 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

9.1.2 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`

9.2 texture.t2d.Loader

9.2.1 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

9.2.2 Attributes

`Loader.kind = '2d'`

`Loader.file_extensions = []`

`Loader.ctx`

ModernGL context

Type `moderngl.Context`

9.3 program.single.Loader

9.3.1 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

9.3.2 Attributes

Loader.**kind** = 'single'

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

```
Loader.ctx
```

ModernGL context

Type moderngl.Context

9.4 program.separate.Loader

9.4.1 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

9.4.2 Loader Specific Methods

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

9.4.3 Attributes

`Loader.kind = 'separate'`

`Loader.file_extensions = []`

`Loader.ctx`

ModernGL context

Type moderngl.Context

9.5 texture.array.Loader

9.5.1 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

9.5.2 Attributes

`Loader.kind = 'array'`

`Loader.file_extensions = []`

`Loader.ctx`

ModernGL context

Type `moderngl.Context`

9.6 scene.wavefront.Loader

9.6.1 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

9.6.2 Attributes

`Loader.kind = 'wavefront'`

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

`Loader.ctx`

ModernGL context

Type `moderngl.Context`

9.7 scene.gltf2.Loader

9.7.1 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

9.7.2 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

9.7.3 Attributes

`Loader.kind = 'gltf'`

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

`Loader.ctx`

ModernGL context

Type `moderngl.Context`

9.7.4 Loader Specific Attributes

`Loader.supported_extensions = []`

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

9.8 scene.stl.Loader

9.8.1 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

9.8.2 Attributes

`Loader.kind = 'stl'`

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

`Loader.ctx`

ModernGL context

Type `moderngl.Context`

9.9 data.json.Loader

9.9.1 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

9.9.2 Attributes

`Loader.kind = 'json'`

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

`Loader.ctx`

ModernGL context

Type `moderngl.Context`

9.10 data.text.Loader

9.10.1 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

9.10.2 Attributes

`Loader.kind = 'text'`

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

`Loader.ctx`

ModernGL context

Type moderngl.Context

9.11 data.binary.Loader

9.11.1 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

9.11.2 Attributes

`Loader.kind = 'binary'`

`Loader.file_extensions = []`

`Loader.ctx`

ModernGL context

Type `moderngl.Context`

MODERNGL_WINDOW.META

10.1 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.

10.1.1 Methods

`ResourceDescription.__init__ (**kwargs)`

Initialize a resource description

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

10.1.2 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

10.2 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')
```

10.2.1 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 horisontally
- **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

10.2.2 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

10.2.3 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'`

10.3 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',
)
```

10.3.1 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 fragment shader
- **tess_control_shader** (*str*) –
- **tess_evaluation_shader** (*str*) – Path to tess eval shader
- ****kwargs** – Optional custom attributes

10.3.2 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*

10.3.3 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 where a loader class is assigned based on the *kind*.

Type Type

`ProgramDescription.default_kind = None`

`ProgramDescription.resource_type = 'programs'`

10.4 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.

10.4.1 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

10.4.2 Attributes

`SceneDescription.attr_names`

Attribute name config

Type `AttributeNames`

`SceneDescription.cache`

Use cache feature in scene loader

Type `bool`

10.4.3 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 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 where a loader class is assigned based on the *kind*.

Type Type

SceneDescription.**default_kind** = None

SceneDescription.**resource_type** = 'scenes'

10.5 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')
```

10.5.1 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

10.5.2 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 on 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 where a loader class is assigned based on the *kind*.

Type `Type`

`DataDescription.default_kind = None`

`DataDescription.resource_type = 'data'`

MODERNGL_WINDOW.FINDERS

11.1 base.BaseFileSystemFinder

`moderngl_window.finders.base.BaseFileSystemFinder`
Base class for searching filesystem directories

11.1.1 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

11.1.2 Attributes

`BaseFileSystemFinder.settings_attr = None`
Name of the attribute in *Settings* containing a list of paths the finder should search in.
Type str

11.2 texture.FileSystemFinder

`moderngl_window.finders.texture.FileSystemFinder`
Find textures in `settings.TEXTURE_DIRS`

11.2.1 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

11.2.2 Attributes

`FileSystemFinder.settings_attr = 'TEXTURE_DIRS'`

11.3 program.FileSystemFinder

`moderngl_window.finders.program.FileSystemFinder`
Find shaders in `settings.PROGRAM_DIRS`

11.3.1 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

11.3.2 Attributes

`FileSystemFinder.settings_attr = 'PROGRAM_DIRS'`

11.4 scene.FileSystemFinder

`moderngl_window.finders.scene.FileSystemFinder`
Find scenes in `settings.SCENE_DIRS`

11.4.1 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

11.4.2 Attributes

`FileSystemFinder.settings_attr = 'SCENE_DIRS'`

11.5 data.FilesystemFinder

moderngl_window.finders.data.**FilesystemFinder**

Find data in settings.DATA_DIRS

11.5.1 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

11.5.2 Attributes

FilesystemFinder.**settings_attr** = 'DATA_DIRS'

MODERNGL_WINDOW.OPENGL

12.1 opengl.projection.Projection3D

`moderngl_window.opengl.projection.Projection3D`
3D Projection

12.1.1 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*) – Sspect 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*) – Sspect 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

12.1.2 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.

12.2 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;
```

12.2.1 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`

12.2.2 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

13.1 base.BaseRegistry

`moderngl_window.resources.base.BaseRegistry`

Base class for all resource pools

13.1.1 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, resoure) 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

13.1.2 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

13.2 textures.Textures

`moderngl_window.resources.textures.Textures`

Handles texture resources

13.2.1 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, resoure) tuples

`Textures.resolve_loader(meta: moderngl_window.meta.base.ResourceDescription)` → None
 Attempts to assign a loader class to a ResourceDecription.

Parameters `meta` (*ResourceDescription*) – The resource description instance

13.2.2 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

13.3 programs.Programs

`moderngl_window.resources.programs.Programs`

Handle program loading

13.3.1 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, resoure) 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

13.3.2 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

13.4 scenes.Scenes

`moderngl_window.resources.scenes.Scenes`

Handles scene loading

13.4.1 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, resoure) tuples

`Scenes.resolve_loader` (*meta: moderngl_window.meta.base.ResourceDescription*) → None

Attempts to assign a loader class to a ResourceDecription.

Parameters *meta* (*ResourceDescription*) – The resource description instance

13.4.2 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

13.5 base.DataFiles

`moderngl_window.resources.data.DataFiles`

Registry for requested data files

13.5.1 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, resoure) tuples

`DataFiles.resolve_loader(meta: moderngl_window.meta.base.ResourceDescription) → None`

Attempts to assign a loader class to a ResourceDecription.

Parameters `meta` (*ResourceDescription*) – The resource description instance

13.5.2 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

14.1 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.

14.1.1 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

14.1.2 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 the current time in seconds

The current time can also be assigned to this attribute.

Returns The current time in seconds

Return type float

14.2 clock.Timer

`moderngl_window.timers.clock.Timer`

Timer based on python time.

14.2.1 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]

14.2.2 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 the current time in seconds

Returns The current time in seconds

MODERNGL_WINDOW.SCENE

15.1 Camera

`moderngl_window.scene.Camera`
Simple camera class containing projection

15.1.1 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`

15.1.2 Attributes

Camera.**matrix**

The current view matrix for the camera

Type numpy.ndarray

15.2 KeyboardCamera

moderngl_window.scene.**KeyboardCamera**

Camera controlled by mouse and keyboard

15.2.1 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.**move_left** (*activate*) → None

The camera should be continiously moving to the left.

Parameters **activate** (*bool*) – Activate or deactivate this state

KeyboardCamera.**move_right** (*activate*) → None

The camera should be continiously moving to the right.

Parameters **activate** (*bool*) – Activate or deactivate this state

KeyboardCamera.**move_forward** (*activate*) → None

The camera should be continiously moving forward.

Parameters **activate** (*bool*) – Activate or deactivate this state

KeyboardCamera.**move_backward** (*activate*) → None

The camera should be continiously 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**(*x, y*) → None

Set the rotation state of the camera. This value is normally the current mouse position.

Parameters

- **x** – viewport x pos
- **y** – viewport y pos

15.2.2 Inherited Methods

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

15.2.3 Attributes

KeyboardCamera.**matrix**

The current view matrix for the camera

Type numpy.ndarray

15.3 Scene

15.3.1 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 ()` assining default meshprograms if needed and sets the model matrix.

`Scene.destroy () → None`

Destroys the scene data and vertex buffers

15.3.2 Attributes

`Scene.ctx`

The current context

Type `moderngl.Context`

`Scene.model_matrix`

The current model matrix

This property is settable.

Type `numpy.ndarray`

15.4 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.

15.4.1 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

15.4.2 Attributes

`Node.children`
List of children
Type list

15.5 Mesh

`moderngl_window.scene.Mesh = <class 'moderngl_window.scene.mesh.Mesh'>`
Mesh info and geometry

15.5.1 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** (*Msterial*) – 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

15.6 Material

`moderngl_window.scene.Material`

Generic material

15.6.1 Methods

`Material.__init__(name)`

Initialize material.

Parameters **name** (*str*) – Name of the material

15.6.2 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

15.7 MaterialTexture

`moderngl_window.scene.MaterialTexture`
 Wrapper for textures used in materials. Contains a texture and a sampler object.

15.7.1 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

15.7.2 Attributes

`MaterialTexture.texture`
 Texture instance
Type `moderngl.Texture`

`MaterialTexture.sampler`
 Sampler instance
Type `moderngl.Sampler`

15.8 MeshProgram

`moderngl_window.scene.MeshProgram`
 Describes how a mesh is rendered using a specific shader program

15.8.1 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

15.8.2 Attributes

`MeshProgram.ctx`

The current context

Type `moderngl.Context`

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

m

`moderngl_window`, 13
`moderngl_window.conf`, 16
`moderngl_window.context.base.window`, 23
`moderngl_window.context.glfw.window`, 32
`moderngl_window.context.headless.window`, 37
`moderngl_window.context.pyglet.window`, 42
`moderngl_window.context.pyqt5.window`, 47
`moderngl_window.context.pyside2.window`, 52
`moderngl_window.context.sdl2.window`, 57
`moderngl_window.finders.base`, 89
`moderngl_window.finders.data`, 90
`moderngl_window.finders.program`, 90
`moderngl_window.finders.scene`, 90
`moderngl_window.finders.texture`, 89
`moderngl_window.geometry`, 61
`moderngl_window.loaders.base`, 65
`moderngl_window.loaders.data.binary`, 76
`moderngl_window.loaders.data.json`, 74
`moderngl_window.loaders.data.text`, 75
`moderngl_window.loaders.program.separate`, 69
`moderngl_window.loaders.program.single`, 67
`moderngl_window.loaders.scene.glTF2`, 72
`moderngl_window.loaders.scene.stl`, 73
`moderngl_window.loaders.scene.wavefront`, 71
`moderngl_window.loaders.texture.array`, 70
`moderngl_window.loaders.texture.t2d`, 66
`moderngl_window.meta.base`, 79
`moderngl_window.meta.data`, 86
`moderngl_window.meta.program`, 82
`moderngl_window.meta.scene`, 84
`moderngl_window.meta.texture`, 80
`moderngl_window.opengl.projection`, 93
`moderngl_window.opengl.vao`, 94
`moderngl_window.resources`, 96
`moderngl_window.resources.base`, 97
`moderngl_window.resources.data`, 101
`moderngl_window.resources.programs`, 99
`moderngl_window.resources.scenes`, 100
`moderngl_window.resources.textures`, 98
`moderngl_window.scene`, 109
`moderngl_window.timers.base`, 103
`moderngl_window.timers.clock`, 104

INDEX

Symbols

`__init__()` (`moderngl_window.conf.Settings` method), 17
`__init__()` (`moderngl_window.context.base.window.BaseWindow` method), 28
`__init__()` (`moderngl_window.context.base.window.WindowConfig` method), 24
`__init__()` (`moderngl_window.context.glfw.window.Window` method), 32
`__init__()` (`moderngl_window.context.headless.window.Window` method), 38
`__init__()` (`moderngl_window.context.pyglet.window.Window` method), 42
`__init__()` (`moderngl_window.context.pyqt5.window.Window` method), 47
`__init__()` (`moderngl_window.context.pyside2.window.Window` method), 52
`__init__()` (`moderngl_window.context.sdl2.window.Window` method), 57
`__init__()` (`moderngl_window.finders.base.BaseFilesystemFinder` method), 89
`__init__()` (`moderngl_window.finders.data.FilesystemFinder` method), 91
`__init__()` (`moderngl_window.finders.program.FilesystemFinder` method), 90
`__init__()` (`moderngl_window.finders.scene.FilesystemFinder` method), 90
`__init__()` (`moderngl_window.finders.texture.FilesystemFinder` method), 89
`__init__()` (`moderngl_window.loaders.base.BaseLoader` method), 65
`__init__()` (`moderngl_window.loaders.data.binary.Loader` method), 77
`__init__()` (`moderngl_window.loaders.data.json.Loader` method), 75
`__init__()` (`moderngl_window.loaders.data.text.Loader` method), 76
`__init__()` (`moderngl_window.loaders.program.separate.Loader` method), 69
`__init__()` (`moderngl_window.loaders.program.single.Loader` method), 67
`__init__()` (`moderngl_window.loaders.scene.gltf2.Loader` method), 72
`__init__()` (`moderngl_window.loaders.scene.stl.Loader` method), 74
`__init__()` (`moderngl_window.loaders.scene.wavefront.Loader` method), 71
`__init__()` (`moderngl_window.loaders.texture.array.Loader` method), 70
`__init__()` (`moderngl_window.loaders.texture.t2d.Loader` method), 66
`__init__()` (`moderngl_window.meta.base.ResourceDescription` method), 79
`__init__()` (`moderngl_window.meta.data.DataDescription` method), 87
`__init__()` (`moderngl_window.meta.program.ProgramDescription` method), 83
`__init__()` (`moderngl_window.meta.scene.SceneDescription` method), 85
`__init__()` (`moderngl_window.meta.texture.TextureDescription` method), 80
`__init__()` (`moderngl_window.opengl.projection.Projection3D` method), 93
`__init__()` (`moderngl_window.opengl.vao.VAO` method), 95
`__init__()` (`moderngl_window.resources.base.BaseRegistry` method), 97
`__init__()` (`moderngl_window.resources.data.DataFiles` method), 101
`__init__()` (`moderngl_window.resources.programs.Programs` method), 99
`__init__()` (`moderngl_window.resources.scenes.Scenes` method), 100
`__init__()` (`moderngl_window.resources.textures.Textures` method), 98
`__init__()` (`moderngl_window.scene.Camera` method), 107
`__init__()` (`moderngl_window.scene.KeyboardCamera` method), 108
`__init__()` (`moderngl_window.scene.Material` method), 113
`__init__()` (`moderngl_window.scene.MaterialTexture` method), 114
`__init__()` (`moderngl_window.scene.Mesh` method),

112
 __init__() (moderngl_window.scene.MeshProgram
 method), 114
 __init__() (moderngl_window.scene.Node method),
 111
 __init__() (moderngl_window.scene.Scene method),
 110
 __init__() (moderngl_window.timers.base.BaseTimer
 method), 103
 __init__() (moderngl_window.timers.clock.Timer
 method), 104

A

activate_context() (in module mod-
 erngl_window), 15
 add() (moderngl_window.resources.base.BaseRegistry
 method), 97
 add() (moderngl_window.resources.data.DataFiles
 method), 101
 add() (moderngl_window.resources.programs.Programs
 method), 99
 add() (moderngl_window.resources.scenes.Scenes
 method), 100
 add() (moderngl_window.resources.textures.Textures
 method), 98
 add_attribute() (moderngl_window.scene.Mesh
 method), 113
 add_child() (moderngl_window.scene.Node
 method), 111
 anisotropy (moderngl_window.meta.texture.TextureDescription
 attribute), 81
 apply() (moderngl_window.scene.MeshProgram
 method), 115
 apply_default_settings() (mod-
 erngl_window.conf.Settings method), 17
 apply_from_cls() (moderngl_window.conf.Settings
 method), 18
 apply_from_dict() (mod-
 erngl_window.conf.Settings method), 18
 apply_from_iterable() (mod-
 erngl_window.conf.Settings method), 18
 apply_from_module() (mod-
 erngl_window.conf.Settings method), 18
 apply_from_module_name() (mod-
 erngl_window.conf.Settings method), 18
 apply_mesh_programs() (mod-
 erngl_window.scene.Scene method), 110
 apply_settings_from_env() (mod-
 erngl_window.conf.Settings method), 17
 aspect_ratio (mod-
 erngl_window.context.base.window.BaseWindow
 attribute), 31
 aspect_ratio (mod-
 erngl_window.context.base.window.WindowConfig

attribute), 27
 aspect_ratio (mod-
 erngl_window.context.glfw.window.Window
 attribute), 36
 aspect_ratio (mod-
 erngl_window.context.headless.window.Window
 attribute), 40
 aspect_ratio (mod-
 erngl_window.context.pyglet.window.Window
 attribute), 45
 aspect_ratio (mod-
 erngl_window.context.pyqt5.window.Window
 attribute), 50
 aspect_ratio (mod-
 erngl_window.context.pyside2.window.Window
 attribute), 55
 aspect_ratio (mod-
 erngl_window.context.sdl2.window.Window
 attribute), 60
 aspect_ratio (mod-
 erngl_window.opengl.projection.Projection3D
 attribute), 93
 attr_names (moderngl_window.meta.scene.SceneDescription
 attribute), 85
 attrs (moderngl_window.meta.base.ResourceDescription
 attribute), 79
 attrs (moderngl_window.meta.data.DataDescription
 attribute), 87
 attrs (moderngl_window.meta.program.ProgramDescription
 attribute), 84
 attrs (moderngl_window.meta.scene.SceneDescription
 attribute), 85
 attrs (moderngl_window.meta.texture.TextureDescription
 attribute), 81

B

BaseFilesystemFinder (in module mod-
 erngl_window.finders.base), 89
 BaseRegistry (in module mod-
 erngl_window.resources.base), 97
 BaseTimer (in module moderngl_window.timers.base),
 103
 bbox() (in module moderngl_window.geometry), 63
 buffer() (moderngl_window.opengl.vao.VAO
 method), 96
 buffer_size (moderngl_window.context.base.window.BaseWindow
 attribute), 30
 buffer_size (moderngl_window.context.glfw.window.Window
 attribute), 35
 buffer_size (moderngl_window.context.headless.window.Window
 attribute), 39
 buffer_size (moderngl_window.context.pyglet.window.Window
 attribute), 45

- [attribute](#)), 69
- ctx (moderngl_window.loaders.scene.gltf2.Loader attribute), 73
- ctx (moderngl_window.loaders.scene.stl.Loader attribute), 74
- ctx (moderngl_window.loaders.scene.wavefront.Loader attribute), 72
- ctx (moderngl_window.loaders.texture.array.Loader attribute), 71
- ctx (moderngl_window.loaders.texture.t2d.Loader attribute), 67
- ctx (moderngl_window.opengl.vao.VAO attribute), 96
- ctx (moderngl_window.scene.MeshProgram attribute), 115
- ctx (moderngl_window.scene.Scene attribute), 110
- ctx () (in module moderngl_window), 15
- cube () (in module moderngl_window.geometry), 64
- cursor (moderngl_window.context.base.window.BaseWindow attribute), 31
- cursor (moderngl_window.context.base.window.WindowConfig attribute), 28
- cursor (moderngl_window.context.glfw.window.Window attribute), 36
- cursor (moderngl_window.context.headless.window.Window attribute), 40
- cursor (moderngl_window.context.pyglet.window.Window attribute), 46
- cursor (moderngl_window.context.pyqt5.window.Window attribute), 51
- cursor (moderngl_window.context.pyside2.window.Window attribute), 56
- cursor (moderngl_window.context.sdl2.window.Window attribute), 60
- D**
- DATA_DIRS (moderngl_window.conf.Settings attribute), 20
- DATA_FINDERS (moderngl_window.conf.Settings attribute), 20
- DATA_LOADERS (moderngl_window.conf.Settings attribute), 21
- DataDescription (in module moderngl_window.meta.data), 86
- DataFiles (in module moderngl_window.resources.data), 101
- default_kind (moderngl_window.meta.base.ResourceDescription attribute), 80
- default_kind (moderngl_window.meta.data.DataDescription attribute), 87
- default_kind (moderngl_window.meta.program.ProgramDescription attribute), 84
- default_kind (moderngl_window.meta.scene.SceneDescription attribute), 86
- default_kind (moderngl_window.meta.texture.TextureDescription attribute), 82
- destroy () (moderngl_window.context.base.window.BaseWindow method), 29
- destroy () (moderngl_window.context.glfw.window.Window method), 33
- destroy () (moderngl_window.context.headless.window.Window method), 39
- destroy () (moderngl_window.context.pyglet.window.Window method), 43
- destroy () (moderngl_window.context.pyqt5.window.Window method), 48
- destroy () (moderngl_window.context.pyside2.window.Window method), 53
- destroy () (moderngl_window.context.sdl2.window.Window method), 58
- destroy () (moderngl_window.scene.Scene method), 110
- double_sided (moderngl_window.scene.Material attribute), 114
- draw () (moderngl_window.scene.Mesh method), 112
- draw () (moderngl_window.scene.MeshProgram method), 114
- draw () (moderngl_window.scene.Node method), 111
- draw () (moderngl_window.scene.Scene method), 110
- draw_bbox () (moderngl_window.scene.Mesh method), 112
- draw_bbox () (moderngl_window.scene.Node method), 111
- draw_bbox () (moderngl_window.scene.Scene method), 110
- F**
- far (moderngl_window.opengl.projection.Projection3D attribute), 94
- fbo (moderngl_window.context.base.window.BaseWindow attribute), 30
- fbo (moderngl_window.context.glfw.window.Window attribute), 35
- fbo (moderngl_window.context.headless.window.Window attribute), 39
- fbo (moderngl_window.context.pyglet.window.Window attribute), 44
- fbo (moderngl_window.context.pyqt5.window.Window attribute), 49
- fbo (moderngl_window.context.pyside2.window.Window attribute), 54
- fbo (moderngl_window.context.sdl2.window.Window attribute), 59

file_extensions	(moderngl_window.loaders.base.BaseLoader attribute), 66	method), 77
file_extensions	(moderngl_window.loaders.data.binary.Loader attribute), 77	find_data() (moderngl_window.loaders.data.json.Loader method), 75
file_extensions	(moderngl_window.loaders.data.json.Loader attribute), 75	find_data() (moderngl_window.loaders.data.text.Loader method), 76
file_extensions	(moderngl_window.loaders.data.text.Loader attribute), 76	find_data() (moderngl_window.loaders.program.separate.Loader method), 69
file_extensions	(moderngl_window.loaders.program.separate.Loader attribute), 70	find_data() (moderngl_window.loaders.program.single.Loader method), 68
file_extensions	(moderngl_window.loaders.program.single.Loader attribute), 68	find_data() (moderngl_window.loaders.scene.gltf2.Loader method), 72
file_extensions	(moderngl_window.loaders.scene.gltf2.Loader attribute), 73	find_data() (moderngl_window.loaders.scene.stl.Loader method), 74
file_extensions	(moderngl_window.loaders.scene.stl.Loader attribute), 74	find_data() (moderngl_window.loaders.scene.wavefront.Loader method), 71
file_extensions	(moderngl_window.loaders.scene.wavefront.Loader attribute), 72	find_data() (moderngl_window.loaders.texture.array.Loader method), 70
file_extensions	(moderngl_window.loaders.texture.array.Loader attribute), 71	find_data() (moderngl_window.loaders.texture.t2d.Loader method), 66
file_extensions	(moderngl_window.loaders.texture.t2d.Loader attribute), 67	find_program() (moderngl_window.loaders.base.BaseLoader method), 65
FilesystemFinder	(in module moderngl_window.finders.data), 91	find_program() (moderngl_window.loaders.data.binary.Loader method), 77
FilesystemFinder	(in module moderngl_window.finders.program), 90	find_program() (moderngl_window.loaders.data.json.Loader method), 75
FilesystemFinder	(in module moderngl_window.finders.scene), 90	find_program() (moderngl_window.loaders.data.text.Loader method), 76
FilesystemFinder	(in module moderngl_window.finders.texture), 89	find_program() (moderngl_window.loaders.program.separate.Loader method), 69
find()	(moderngl_window.finders.base.BaseFilesystemFinder method), 89	find_program() (moderngl_window.loaders.program.single.Loader method), 68
find()	(moderngl_window.finders.data.FilesystemFinder method), 91	find_program() (moderngl_window.loaders.scene.gltf2.Loader method), 72
find()	(moderngl_window.finders.program.FilesystemFinder method), 90	find_program() (moderngl_window.loaders.scene.stl.Loader method), 74
find()	(moderngl_window.finders.scene.FilesystemFinder method), 90	find_program() (moderngl_window.loaders.scene.wavefront.Loader method), 71
find()	(moderngl_window.finders.texture.FilesystemFinder method), 89	find_program() (moderngl_window.loaders.texture.array.Loader method), 70
find_data()	(moderngl_window.loaders.base.BaseLoader method), 65	find_program() (moderngl_window.loaders.texture.t2d.Loader method), 66
find_data()	(moderngl_window.loaders.data.binary.Loader method), 77	find_scene() (moderngl_window.loaders.base.BaseLoader method), 65

<i>method</i>), 65		<i>method</i>), 74	
find_scene()	(mod- erngl_window.loaders.data.binary.Loader method), 77	find_texture()	(mod- erngl_window.loaders.scene.wavefront.Loader method), 71
find_scene()	(mod- erngl_window.loaders.data.json.Loader method), 75	find_texture()	(mod- erngl_window.loaders.texture.array.Loader method), 70
find_scene()	(mod- erngl_window.loaders.data.text.Loader method), 76	find_texture()	(mod- erngl_window.loaders.texture.t2d.Loader method), 67
find_scene()	(mod- erngl_window.loaders.program.separate.Loader method), 69	find_window_classes()	(in module mod- erngl_window), 15
find_scene()	(mod- erngl_window.loaders.program.single.Loader method), 68	flip(moderngl_window.meta.texture.TextureDescription attribute), 81	
find_scene()	(mod- erngl_window.loaders.scene.glTF2.Loader method), 73	fov(moderngl_window.opengl.projection.Projection3D attribute), 94	
find_scene()	(mod- erngl_window.loaders.scene.stl.Loader method), 74	fragment_shader	(mod- erngl_window.meta.program.ProgramDescription attribute), 83
find_scene()	(mod- erngl_window.loaders.scene.wavefront.Loader method), 72	frames(moderngl_window.context.base.window.BaseWindow attribute), 30	
find_scene()	(mod- erngl_window.loaders.texture.array.Loader method), 70	frames(moderngl_window.context.glfw.window.Window attribute), 35	
find_scene()	(mod- erngl_window.loaders.texture.t2d.Loader method), 67	frames(moderngl_window.context.headless.window.Window attribute), 40	
find_texture()	(mod- erngl_window.loaders.base.BaseLoader method), 65	frames(moderngl_window.context.pyglet.window.Window attribute), 45	
find_texture()	(mod- erngl_window.loaders.data.binary.Loader method), 77	frames(moderngl_window.context.pyqt5.window.Window attribute), 50	
find_texture()	(mod- erngl_window.loaders.data.json.Loader method), 75	frames(moderngl_window.context.pyside2.window.Window attribute), 55	
find_texture()	(mod- erngl_window.loaders.data.text.Loader method), 76	frames(moderngl_window.context.sdl2.window.Window attribute), 59	
find_texture()	(mod- erngl_window.loaders.program.separate.Loader method), 69	fullscreen(moderngl_window.context.base.window.BaseWindow attribute), 30	
find_texture()	(mod- erngl_window.loaders.program.single.Loader method), 68	fullscreen(moderngl_window.context.glfw.window.Window attribute), 36	
find_texture()	(mod- erngl_window.loaders.scene.glTF2.Loader method), 72	fullscreen(moderngl_window.context.headless.window.Window attribute), 40	
find_texture()	(mod- erngl_window.loaders.scene.stl.Loader	fullscreen(moderngl_window.context.pyglet.window.Window attribute), 45	
		fullscreen(moderngl_window.context.pyqt5.window.Window attribute), 50	
		fullscreen(moderngl_window.context.pyside2.window.Window attribute), 55	
		fullscreen(moderngl_window.context.sdl2.window.Window attribute), 60	
		G	
		geometry_shader	(mod- erngl_window.meta.program.ProgramDescription attribute), 83
		get_buffer_by_name()	(mod- erngl_window.opengl.vao.VAO method),

96
get_local_window_cls() (in module moderngl_window), 15
get_window_cls() (in module moderngl_window), 15
gl_version(moderngl_window.context.base.window.BaseWindow attribute), 30
gl_version(moderngl_window.context.base.window.Window attribute), 27
gl_version(moderngl_window.context.glfw.window.Window attribute), 35
gl_version(moderngl_window.context.headless.window.Window attribute), 39
gl_version(moderngl_window.context.pyglet.window.Window attribute), 45
gl_version(moderngl_window.context.pyqt5.window.Window attribute), 50
gl_version(moderngl_window.context.pyside2.window.Window attribute), 55
gl_version(moderngl_window.context.sdl2.window.Window attribute), 59
gl_version_code(moderngl_window.context.base.window.BaseWindow attribute), 32
gl_version_code(moderngl_window.context.glfw.window.Window attribute), 37
gl_version_code(moderngl_window.context.headless.window.Window attribute), 41
gl_version_code(moderngl_window.context.pyglet.window.Window attribute), 47
gl_version_code(moderngl_window.context.pyqt5.window.Window attribute), 52
gl_version_code(moderngl_window.context.pyside2.window.Window attribute), 57
gl_version_code(moderngl_window.context.sdl2.window.Window attribute), 61
glfw_char_callback() (moderngl_window.context.glfw.window.Window method), 34
glfw_key_event_callback() (moderngl_window.context.glfw.window.Window method), 34
glfw_mouse_button_callback() (moderngl_window.context.glfw.window.Window method), 34
glfw_mouse_event_callback() (moderngl_window.context.glfw.window.Window method), 34
glfw_mouse_scroll_callback() (moderngl_window.context.glfw.window.Window method), 34
glfw_window_resize_callback() (moderngl_window.context.glfw.window.Window method), 34
height(moderngl_window.context.base.window.BaseWindow attribute), 30
height(moderngl_window.context.glfw.window.Window attribute), 35
height(moderngl_window.context.headless.window.Window attribute), 39
height(moderngl_window.context.pyglet.window.Window attribute), 45
height(moderngl_window.context.pyqt5.window.Window attribute), 50
height(moderngl_window.context.pyside2.window.Window attribute), 55
height(moderngl_window.context.sdl2.window.Window attribute), 59
has_normals() (moderngl_window.scene.Mesh method), 113
has_uvs() (moderngl_window.scene.Mesh method), 113
image(moderngl_window.meta.texture.TextureDescription attribute), 81
index_buffer() (moderngl_window.opengl.vao.VAO method), 96
init_mgl_context() (moderngl_window.context.base.window.BaseWindow method), 28
init_mgl_context() (moderngl_window.context.glfw.window.Window method), 33
init_mgl_context() (moderngl_window.context.headless.window.Window method), 38
init_mgl_context() (moderngl_window.context.pyglet.window.Window method), 42
init_mgl_context() (moderngl_window.context.pyqt5.window.Window method), 47
init_mgl_context() (moderngl_window.context.pyside2.window.Window method), 52
init_mgl_context() (moderngl_window.context.sdl2.window.Window method), 57

instance() (moderngl_window.opengl.vao.VAO attribute), 36
 method), 96 key_event_func (moderngl_window.context.base.window.BaseWindow attribute), 32
 is_closing(moderngl_window.context.glfw.window.Window attribute), 37
 is_closing(moderngl_window.context.headless.window.Window attribute), 41
 is_closing(moderngl_window.context.pyglet.window.Window attribute), 46
 is_closing(moderngl_window.context.pyqt5.window.Window attribute), 51
 is_closing(moderngl_window.context.pyside2.window.Window attribute), 56
 is_closing(moderngl_window.context.sdl2.window.Window attribute), 60
 is_key_pressed() (moderngl_window.context.base.window.BaseWindow method), 29
 is_key_pressed() (moderngl_window.context.glfw.window.Window method), 33
 is_key_pressed() (moderngl_window.context.headless.window.Window method), 38
 is_key_pressed() (moderngl_window.context.pyglet.window.Window method), 42
 is_key_pressed() (moderngl_window.context.pyqt5.window.Window method), 48
 is_key_pressed() (moderngl_window.context.pyside2.window.Window method), 53
 is_key_pressed() (moderngl_window.context.sdl2.window.Window method), 58
 is_paused(moderngl_window.timers.base.BaseTimer attribute), 103
 is_paused(moderngl_window.timers.clock.Timer attribute), 104
 is_running(moderngl_window.timers.base.BaseTimer attribute), 103
 is_running(moderngl_window.timers.clock.Timer attribute), 104

K

key_event() (moderngl_window.context.base.window.Window method), 24
 key_event_func (moderngl_window.context.base.window.BaseWindow attribute), 31
 key_event_func (moderngl_window.context.glfw.window.Window attribute), 36
 key_event_func (moderngl_window.context.headless.window.Window attribute), 40
 key_event_func (moderngl_window.context.pyglet.window.Window attribute), 46
 key_event_func (moderngl_window.context.pyqt5.window.Window attribute), 51
 key_event_func (moderngl_window.context.pyside2.window.Window attribute), 56
 key_event_func (moderngl_window.context.sdl2.window.Window attribute), 60
 key_input() (moderngl_window.scene.KeyboardCamera method), 108
 key_pressed_event() (moderngl_window.context.pyqt5.window.Window method), 49
 key_pressed_event() (moderngl_window.context.pyside2.window.Window method), 54
 key_release_event() (moderngl_window.context.pyqt5.window.Window method), 49
 key_release_event() (moderngl_window.context.pyside2.window.Window method), 54
 KeyboardCamera (in module moderngl_window.scene), 108
 keys(moderngl_window.context.base.window.BaseWindow attribute), 29
 keys(moderngl_window.context.glfw.window.Window attribute), 35
 keys(moderngl_window.context.headless.window.Window attribute), 39
 keys(moderngl_window.context.pyglet.window.Window attribute), 44
 keys(moderngl_window.context.pyqt5.window.Window attribute), 49
 keys(moderngl_window.context.pyside2.window.Window attribute), 54
 keys(moderngl_window.context.sdl2.window.Window attribute), 59
 kind(moderngl_window.loaders.base.BaseLoader attribute), 66
 kind(moderngl_window.loaders.data.binary.Loader attribute), 77
 kind(moderngl_window.loaders.data.json.Loader attribute), 75
 kind(moderngl_window.loaders.data.text.Loader attribute), 76

kind (moderngl_window.loaders.program.separate.Loader attribute), 70	load () (moderngl_window.loaders.scene.wavefront.Loader method), 71
kind (moderngl_window.loaders.program.single.Loader attribute), 68	load () (moderngl_window.loaders.texture.array.Loader method), 70
kind (moderngl_window.loaders.scene.gltf2.Loader attribute), 73	load () (moderngl_window.loaders.texture.t2d.Loader method), 66
kind (moderngl_window.loaders.scene.stl.Loader attribute), 74	load () (moderngl_window.resources.base.BaseRegistry method), 97
kind (moderngl_window.loaders.scene.wavefront.Loader attribute), 72	load () (moderngl_window.resources.data.DataFiles method), 101
kind (moderngl_window.loaders.texture.array.Loader attribute), 71	load () (moderngl_window.resources.programs.Programs method), 99
kind (moderngl_window.loaders.texture.t2d.Loader attribute), 67	load () (moderngl_window.resources.scenes.Scenes method), 100
kind (moderngl_window.meta.base.ResourceDescription attribute), 79	load () (moderngl_window.resources.textures.Textures method), 98
kind (moderngl_window.meta.data.DataDescription attribute), 87	load_binary () (moderngl_window.context.base.window.WindowConfig method), 26
kind (moderngl_window.meta.program.ProgramDescription attribute), 84	load_glb () (moderngl_window.loaders.scene.gltf2.Loader method), 73
kind (moderngl_window.meta.scene.SceneDescription attribute), 86	load_gltf () (moderngl_window.loaders.scene.gltf2.Loader method), 73
kind (moderngl_window.meta.texture.TextureDescription attribute), 82	load_images () (moderngl_window.loaders.scene.gltf2.Loader method), 73
L	
label (moderngl_window.meta.base.ResourceDescription attribute), 79	load_json () (moderngl_window.context.base.window.WindowConfig method), 26
label (moderngl_window.meta.data.DataDescription attribute), 87	load_materials () (moderngl_window.loaders.scene.gltf2.Loader method), 73
label (moderngl_window.meta.program.ProgramDescription attribute), 84	load_meshes () (moderngl_window.loaders.scene.gltf2.Loader method), 73
label (moderngl_window.meta.scene.SceneDescription attribute), 86	load_node () (moderngl_window.loaders.scene.gltf2.Loader method), 73
label (moderngl_window.meta.texture.TextureDescription attribute), 81	load_nodes () (moderngl_window.loaders.scene.gltf2.Loader method), 73
layers (moderngl_window.meta.texture.TextureDescription attribute), 81	load_pool () (moderngl_window.resources.base.BaseRegistry method), 97
load () (moderngl_window.loaders.base.BaseLoader method), 65	load_pool () (moderngl_window.resources.data.DataFiles method), 101
load () (moderngl_window.loaders.data.binary.Loader method), 77	load_pool () (moderngl_window.resources.programs.Programs method), 99
load () (moderngl_window.loaders.data.json.Loader method), 75	load_pool () (moderngl_window.resources.scenes.Scenes method), 100
load () (moderngl_window.loaders.data.text.Loader method), 76	load_pool () (moderngl_window.resources.textures.Textures method), 98
load () (moderngl_window.loaders.program.separate.Loader method), 69	load_program () (moderngl_window.context.base.window.WindowConfig method), 26
load () (moderngl_window.loaders.program.single.Loader method), 67	load_samplers () (moderngl_window.loaders.scene.gltf2.Loader method), 74
load () (moderngl_window.loaders.scene.gltf2.Loader method), 72	
load () (moderngl_window.loaders.scene.stl.Loader method), 74	

[method](#)), 73
[load_scene\(\)](#) ([moderngl_window.context.base.window.WindowConfig](#) [attribute](#)), 108
[load_shader\(\)](#) ([moderngl_window.loaders.program.separate.Loader](#) [attribute](#)), 27
[load_text\(\)](#) ([moderngl_window.context.base.window.WindowConfig](#) [attribute](#)), 26
[load_texture_2d\(\)](#) ([moderngl_window.context.base.window.WindowConfig](#) [attribute](#)), 25
[load_texture_array\(\)](#) ([moderngl_window.context.base.window.WindowConfig](#) [attribute](#)), 25
[load_textures\(\)](#) ([moderngl_window.loaders.scene.glTF2.Loader](#) [attribute](#)), 73
[loader_cls\(\)](#) ([moderngl_window.meta.base.ResourceDescription](#) [attribute](#)), 80
[loader_cls\(\)](#) ([moderngl_window.meta.data.DataDescription](#) [attribute](#)), 87
[loader_cls\(\)](#) ([moderngl_window.meta.program.ProgramDescription](#) [attribute](#)), 84
[loader_cls\(\)](#) ([moderngl_window.meta.scene.SceneDescription](#) [attribute](#)), 86
[loader_cls\(\)](#) ([moderngl_window.meta.texture.TextureDescription](#) [attribute](#)), 82
[loaders\(\)](#) ([moderngl_window.resources.base.BaseRegistry](#) [attribute](#)), 98
[loaders\(\)](#) ([moderngl_window.resources.data.DataFiles](#) [attribute](#)), 102
[loaders\(\)](#) ([moderngl_window.resources.programs.Programs](#) [attribute](#)), 100
[loaders\(\)](#) ([moderngl_window.resources.scenes.Scenes](#) [attribute](#)), 101
[loaders\(\)](#) ([moderngl_window.resources.textures.Textures](#) [attribute](#)), 99
[log_level\(\)](#) ([moderngl_window.context.base.window.WindowConfig](#) [attribute](#)), 28
[look_at\(\)](#) ([moderngl_window.scene.Camera](#) [method](#)), 107
[look_at\(\)](#) ([moderngl_window.scene.KeyboardCamera](#) [method](#)), 109
[matrix](#) ([moderngl_window.scene.Camera](#) [attribute](#)), 108
[matrix](#) ([moderngl_window.scene.KeyboardCamera](#) [attribute](#)), 109
[Mesh](#) (in module [moderngl_window.scene](#)), 112
[MeshProgram](#) (in module [moderngl_window.scene](#)), 114
[mipmap_levels](#) ([moderngl_window.meta.texture.TextureDescription](#) [attribute](#)), 81
[model_matrix](#) ([moderngl_window.scene.Scene](#) [attribute](#)), 110
[moderngl_window](#) (module), 13
[moderngl_window.conf](#) (module), 16
[moderngl_window.context.base.window](#) (module), 23, 28
[moderngl_window.context.glfw.window](#) (module), 32
[moderngl_window.context.headless.window](#) (module), 37
[moderngl_window.context.pyglet.window](#) (module), 42
[moderngl_window.context.pyqt5.window](#) (module), 47
[moderngl_window.context.pyside2.window](#) (module), 52
[moderngl_window.context.sdl2.window](#) (module), 57
[moderngl_window.finders.base](#) (module), 89
[moderngl_window.finders.data](#) (module), 90
[moderngl_window.finders.program](#) (module), 90
[moderngl_window.finders.scene](#) (module), 90
[moderngl_window.finders.texture](#) (module), 89
[moderngl_window.geometry](#) (module), 61
[moderngl_window.loaders.base](#) (module), 65
[moderngl_window.loaders.data.binary](#) (module), 76
[moderngl_window.loaders.data.json](#) (module), 74
[moderngl_window.loaders.data.text](#) (module), 75
[moderngl_window.loaders.program.separate](#) (module), 69
[moderngl_window.loaders.program.single](#) (module), 67
[moderngl_window.loaders.scene.glTF2](#) (module), 72
[moderngl_window.loaders.scene.stl](#) (module), 73
[moderngl_window.loaders.scene.wavefront](#) (module), 73
[mat_texture](#) ([moderngl_window.scene.Material](#) [attribute](#)), 113
[Material](#) (in module [moderngl_window.scene](#)), 113
[MaterialTexture](#) (in module [moderngl_window.scene](#)), 114
[matrix](#) ([moderngl_window.opengl.projection.Projection3D](#) [attribute](#)), 94

M

- [\(module\), 71](#)
- [moderngl_window.loaders.texture.array \(module\), 70](#)
- [moderngl_window.loaders.texture.t2d \(module\), 66](#)
- [moderngl_window.meta.base \(module\), 79](#)
- [moderngl_window.meta.data \(module\), 86](#)
- [moderngl_window.meta.program \(module\), 82](#)
- [moderngl_window.meta.scene \(module\), 84](#)
- [moderngl_window.meta.texture \(module\), 80](#)
- [moderngl_window.opengl.projection \(module\), 93](#)
- [moderngl_window.opengl.vao \(module\), 94](#)
- [moderngl_window.resources \(module\), 96](#)
- [moderngl_window.resources.base \(module\), 97](#)
- [moderngl_window.resources.data \(module\), 101](#)
- [moderngl_window.resources.programs \(module\), 99](#)
- [moderngl_window.resources.scenes \(module\), 100](#)
- [moderngl_window.resources.textures \(module\), 98](#)
- [moderngl_window.scene \(module\), 107–110, 112–114](#)
- [moderngl_window.timers.base \(module\), 103](#)
- [moderngl_window.timers.clock \(module\), 104](#)
- [modifiers \(moderngl_window.context.base.window.BaseWindow attribute\), 32](#)
- [modifiers \(moderngl_window.context.glfw.window.Window attribute\), 37](#)
- [modifiers \(moderngl_window.context.headless.window.Window attribute\), 41](#)
- [modifiers \(moderngl_window.context.pyglet.window.Window attribute\), 47](#)
- [modifiers \(moderngl_window.context.pyqt5.window.Window attribute\), 52](#)
- [modifiers \(moderngl_window.context.pyside2.window.Window attribute\), 57](#)
- [modifiers \(moderngl_window.context.sdl2.window.Window attribute\), 61](#)
- [mouse \(moderngl_window.context.base.window.BaseWindow attribute\), 32](#)
- [mouse \(moderngl_window.context.glfw.window.Window attribute\), 37](#)
- [mouse \(moderngl_window.context.headless.window.Window attribute\), 41](#)
- [mouse \(moderngl_window.context.pyglet.window.Window attribute\), 47](#)
- [mouse \(moderngl_window.context.pyqt5.window.Window attribute\), 52](#)
- [mouse \(moderngl_window.context.pyside2.window.Window attribute\), 57](#)
- [mouse \(moderngl_window.context.sdl2.window.Window attribute\), 61](#)
- [mouse_drag_event \(\) \(moderngl_window.context.base.window.WindowConfig method\), 25](#)
- [mouse_drag_event_func \(moderngl_window.context.base.window.BaseWindow attribute\), 31](#)
- [mouse_drag_event_func \(moderngl_window.context.glfw.window.Window attribute\), 37](#)
- [mouse_drag_event_func \(moderngl_window.context.headless.window.Window attribute\), 41](#)
- [mouse_drag_event_func \(moderngl_window.context.pyglet.window.Window attribute\), 46](#)
- [mouse_drag_event_func \(moderngl_window.context.pyqt5.window.Window attribute\), 51](#)
- [mouse_drag_event_func \(moderngl_window.context.pyside2.window.Window attribute\), 56](#)
- [mouse_drag_event_func \(moderngl_window.context.sdl2.window.Window attribute\), 61](#)
- [mouse_move_event \(\) \(moderngl_window.context.pyqt5.window.Window method\), 49](#)
- [mouse_move_event \(\) \(moderngl_window.context.pyside2.window.Window method\), 54](#)
- [mouse_position_event \(\) \(moderngl_window.context.base.window.WindowConfig method\), 24](#)
- [mouse_position_event_func \(moderngl_window.context.base.window.BaseWindow attribute\), 31](#)
- [mouse_position_event_func \(moderngl_window.context.glfw.window.Window attribute\), 36](#)
- [mouse_position_event_func \(moderngl_window.context.headless.window.Window attribute\), 40](#)
- [mouse_position_event_func \(moderngl_window.context.pyglet.window.Window attribute\), 46](#)
- [mouse_position_event_func \(moderngl_window.context.pyqt5.window.Window attribute\), 51](#)
- [mouse_position_event_func \(moderngl_window.context.pyside2.window.Window attribute\), 56](#)
- [mouse_position_event_func \(moderngl_window.context.sdl2.window.Window attribute\), 61](#)

<code>erngl_window.context.sdl2.window.Window</code> <code>attribute</code>), 60	<code>erngl_window.context.pyqt5.window.Window</code> <code>attribute</code>), 51
<code>mouse_press_event()</code> (<code>mod-erngl_window.context.base.window.WindowConfig</code> <code>method</code>), 24	<code>mouse_release_event_func</code> (<code>mod-erngl_window.context.pyside2.window.Window</code> <code>attribute</code>), 56
<code>mouse_press_event()</code> (<code>mod-erngl_window.context.pyqt5.window.Window</code> <code>method</code>), 49	<code>mouse_release_event_func</code> (<code>mod-erngl_window.context.sdl2.window.Window</code> <code>attribute</code>), 60
<code>mouse_press_event()</code> (<code>mod-erngl_window.context.pyside2.window.Window</code> <code>method</code>), 54	<code>mouse_scroll_event()</code> (<code>mod-erngl_window.context.base.window.WindowConfig</code> <code>method</code>), 25
<code>mouse_press_event_func</code> (<code>mod-erngl_window.context.base.window.BaseWindow</code> <code>attribute</code>), 31	<code>mouse_scroll_event_func</code> (<code>mod-erngl_window.context.base.window.BaseWindow</code> <code>attribute</code>), 31
<code>mouse_press_event_func</code> (<code>mod-erngl_window.context.glfw.window.Window</code> <code>attribute</code>), 36	<code>mouse_scroll_event_func</code> (<code>mod-erngl_window.context.glfw.window.Window</code> <code>attribute</code>), 37
<code>mouse_press_event_func</code> (<code>mod-erngl_window.context.headless.window.Window</code> <code>attribute</code>), 41	<code>mouse_scroll_event_func</code> (<code>mod-erngl_window.context.headless.window.Window</code> <code>attribute</code>), 41
<code>mouse_press_event_func</code> (<code>mod-erngl_window.context.pyglet.window.Window</code> <code>attribute</code>), 46	<code>mouse_scroll_event_func</code> (<code>mod-erngl_window.context.pyglet.window.Window</code> <code>attribute</code>), 47
<code>mouse_press_event_func</code> (<code>mod-erngl_window.context.pyqt5.window.Window</code> <code>attribute</code>), 51	<code>mouse_scroll_event_func</code> (<code>mod-erngl_window.context.pyqt5.window.Window</code> <code>attribute</code>), 51
<code>mouse_press_event_func</code> (<code>mod-erngl_window.context.pyside2.window.Window</code> <code>attribute</code>), 56	<code>mouse_scroll_event_func</code> (<code>mod-erngl_window.context.pyside2.window.Window</code> <code>attribute</code>), 56
<code>mouse_press_event_func</code> (<code>mod-erngl_window.context.sdl2.window.Window</code> <code>attribute</code>), 60	<code>mouse_scroll_event_func</code> (<code>mod-erngl_window.context.sdl2.window.Window</code> <code>attribute</code>), 61
<code>mouse_release_event()</code> (<code>mod-erngl_window.context.base.window.WindowConfig</code> <code>method</code>), 24	<code>mouse_states</code> (<code>mod-erngl_window.context.base.window.BaseWindow</code> <code>attribute</code>), 32
<code>mouse_release_event()</code> (<code>mod-erngl_window.context.pyqt5.window.Window</code> <code>method</code>), 49	<code>mouse_states</code> (<code>mod-erngl_window.context.glfw.window.Window</code> <code>attribute</code>), 37
<code>mouse_release_event()</code> (<code>mod-erngl_window.context.pyside2.window.Window</code> <code>method</code>), 54	<code>mouse_states</code> (<code>mod-erngl_window.context.headless.window.Window</code> <code>attribute</code>), 41
<code>mouse_release_event_func</code> (<code>mod-erngl_window.context.base.window.BaseWindow</code> <code>attribute</code>), 31	<code>mouse_states</code> (<code>mod-erngl_window.context.pyglet.window.Window</code> <code>attribute</code>), 47
<code>mouse_release_event_func</code> (<code>mod-erngl_window.context.glfw.window.Window</code> <code>attribute</code>), 36	<code>mouse_states</code> (<code>mod-erngl_window.context.pyqt5.window.Window</code> <code>attribute</code>), 52
<code>mouse_release_event_func</code> (<code>mod-erngl_window.context.headless.window.Window</code> <code>attribute</code>), 41	<code>mouse_states</code> (<code>mod-erngl_window.context.pyside2.window.Window</code> <code>attribute</code>), 57
<code>mouse_release_event_func</code> (<code>mod-erngl_window.context.pyglet.window.Window</code> <code>attribute</code>), 46	<code>mouse_states</code> (<code>mod-erngl_window.context.sdl2.window.Window</code> <code>attribute</code>), 61
<code>mouse_release_event_func</code> (<code>mod-</code>	<code>mouse_wheel_event()</code> (<code>mod-</code>

erngl_window.context.pyqt5.window.Window method), 49

mouse_wheel_event() (*moderngl_window.context.pyside2.window.Window* method), 54

move_backward() (*moderngl_window.scene.KeyboardCamera* method), 108

move_down() (*moderngl_window.scene.KeyboardCamera* method), 109

move_forward() (*moderngl_window.scene.KeyboardCamera* method), 108

move_left() (*moderngl_window.scene.KeyboardCamera* method), 108

move_right() (*moderngl_window.scene.KeyboardCamera* method), 108

move_state() (*moderngl_window.scene.KeyboardCamera* method), 109

move_up() (*moderngl_window.scene.KeyboardCamera* method), 108

N

name (*moderngl_window.scene.Material* attribute), 113

near (*moderngl_window.opengl.projection.Projection3D* attribute), 94

next_frame() (*moderngl_window.timers.base.BaseTimer* method), 103

next_frame() (*moderngl_window.timers.clock.Timer* method), 104

Node (in module *moderngl_window.scene*), 111

O

on_key_press() (*moderngl_window.context.pyglet.window.Window* method), 43

on_key_release() (*moderngl_window.context.pyglet.window.Window* method), 43

on_mouse_drag() (*moderngl_window.context.pyglet.window.Window* method), 43

on_mouse_motion() (*moderngl_window.context.pyglet.window.Window* method), 44

on_mouse_press() (*moderngl_window.context.pyglet.window.Window* method), 43

on_mouse_release() (*moderngl_window.context.pyglet.window.Window* method), 44

on_mouse_scroll() (*moderngl_window.context.pyglet.window.Window* method), 44

on_resize() (*moderngl_window.context.pyglet.window.Window* method), 44

on_text() (*moderngl_window.context.pyglet.window.Window* method), 44

P

parse_args() (in module *moderngl_window*), 16

path (*moderngl_window.meta.base.ResourceDescription* attribute), 79

path (*moderngl_window.meta.data.DataDescription* attribute), 87

path (*moderngl_window.meta.program.ProgramDescription* attribute), 84

path (*moderngl_window.meta.scene.SceneDescription* attribute), 85

path (*moderngl_window.meta.texture.TextureDescription* attribute), 81

pause() (*moderngl_window.timers.base.BaseTimer* method), 103

pause() (*moderngl_window.timers.clock.Timer* method), 104

pixel_ratio (*moderngl_window.context.base.window.BaseWindow* attribute), 30

pixel_ratio (*moderngl_window.context.glfw.window.Window* attribute), 35

pixel_ratio (*moderngl_window.context.headless.window.Window* attribute), 39

pixel_ratio (*moderngl_window.context.pyglet.window.Window* attribute), 45

pixel_ratio (*moderngl_window.context.pyqt5.window.Window* attribute), 50

pixel_ratio (*moderngl_window.context.pyside2.window.Window* attribute), 55

pixel_ratio (*moderngl_window.context.sdl2.window.Window* attribute), 59

prepare() (*moderngl_window.scene.Scene* method), 110

print_context_info() (*moderngl_window.context.base.window.BaseWindow* method), 29

print_context_info() (*moderngl_window.context.glfw.window.Window* method), 33

print_context_info() (*moderngl_window.context.headless.window.Window* method), 39

print_context_info() (*moderngl_window.context.pyglet.window.Window* method), 43

print_context_info() (*moderngl_window.context.pyqt5.window.Window* method), 44

method), 48
 print_context_info() (moderngl_window.context.pyside2.window.Window method), 53
 print_context_info() (moderngl_window.context.sdl2.window.Window method), 58
 process_events() (moderngl_window.context.sdl2.window.Window method), 59
 PROGRAM_DIRS (moderngl_window.conf.Settings attribute), 20
 PROGRAM_FINDERS (moderngl_window.conf.Settings attribute), 19
 PROGRAM_LOADERS (moderngl_window.conf.Settings attribute), 20
 ProgramDescription (in module moderngl_window.meta.program), 82
 Programs (in module moderngl_window.resources.programs), 99
 Projection3D (in module moderngl_window.opengl.projection), 93
 projection_constants (moderngl_window.opengl.projection.Projection3D attribute), 94

Q

quad_2d() (in module moderngl_window.geometry), 63
 quad_fs() (in module moderngl_window.geometry), 63

R

register_data_dir() (in module moderngl_window.resources), 97
 register_dir() (in module moderngl_window.resources), 97
 register_program_dir() (in module moderngl_window.resources), 97
 register_scene_dir() (in module moderngl_window.resources), 97
 register_texture_dir() (in module moderngl_window.resources), 97
 release() (moderngl_window.opengl.vao.VAO method), 96
 reloadable (moderngl_window.meta.program.ProgramDescription attribute), 83
 render() (moderngl_window.context.base.window.BaseWindow method), 29
 render() (moderngl_window.context.base.window.WindowConfig method), 24
 render() (moderngl_window.context.glfw.window.Window method), 33

render() (moderngl_window.context.headless.window.Window method), 38
 render() (moderngl_window.context.pyglet.window.Window method), 43
 render() (moderngl_window.context.pyqt5.window.Window method), 48
 render() (moderngl_window.context.pyside2.window.Window method), 53
 render() (moderngl_window.context.sdl2.window.Window method), 58
 render() (moderngl_window.opengl.vao.VAO method), 95
 render_func (moderngl_window.context.base.window.BaseWindow attribute), 31
 render_func (moderngl_window.context.glfw.window.Window attribute), 36
 render_func (moderngl_window.context.headless.window.Window attribute), 40
 render_func (moderngl_window.context.pyglet.window.Window attribute), 46
 render_func (moderngl_window.context.pyqt5.window.Window attribute), 51
 render_func (moderngl_window.context.pyside2.window.Window attribute), 56
 render_func (moderngl_window.context.sdl2.window.Window attribute), 60
 render_indirect() (moderngl_window.opengl.vao.VAO method), 95
 resizable (moderngl_window.context.base.window.BaseWindow attribute), 30
 resizable (moderngl_window.context.base.window.WindowConfig attribute), 27
 resizable (moderngl_window.context.glfw.window.Window attribute), 35
 resizable (moderngl_window.context.headless.window.Window attribute), 40
 resizable (moderngl_window.context.pyglet.window.Window attribute), 45
 resizable (moderngl_window.context.pyqt5.window.Window attribute), 50
 resizable (moderngl_window.context.pyside2.window.Window attribute), 55
 resizable (moderngl_window.context.sdl2.window.Window attribute), 59
 resize() (moderngl_window.context.base.window.BaseWindow method), 29
 resize() (moderngl_window.context.base.window.WindowConfig method), 24
 resize() (moderngl_window.context.glfw.window.Window method), 33
 resize() (moderngl_window.context.headless.window.Window method), 39
 resize() (moderngl_window.context.pyglet.window.Window

[method](#)), 43
[resize\(\)](#) ([moderngl_window.context.pyqt5.window.Window](#)
[method](#)), 48
[resize\(\)](#) ([moderngl_window.context.pyside2.window.Window](#)
[method](#)), 53
[resize\(\)](#) ([moderngl_window.context.sdl2.window.Window](#)
[method](#)), 58
[resize_func\(\)](#) ([moderngl_window.context.base.window.BaseWindow](#)
[attribute](#)), 31
[resize_func\(\)](#) ([moderngl_window.context.glfw.window.Window](#)
[attribute](#)), 36
[resize_func\(\)](#) ([moderngl_window.context.headless.window.Window](#)
[attribute](#)), 40
[resize_func\(\)](#) ([moderngl_window.context.pyglet.window.Window](#)
[attribute](#)), 46
[resize_func\(\)](#) ([moderngl_window.context.pyqt5.window.Window](#)
[attribute](#)), 51
[resize_func\(\)](#) ([moderngl_window.context.pyside2.window.Window](#)
[attribute](#)), 56
[resize_func\(\)](#) ([moderngl_window.context.sdl2.window.Window](#)
[attribute](#)), 60
[resolve_loader\(\)](#) ([moderngl_window.resources.base.BaseRegistry](#)
[method](#)), 98
[resolve_loader\(\)](#) ([moderngl_window.resources.data.DataFiles](#)
[method](#)), 101
[resolve_loader\(\)](#) ([moderngl_window.resources.programs.Programs](#)
[method](#)), 99
[resolve_loader\(\)](#) ([moderngl_window.resources.scenes.Scenes](#)
[method](#)), 100
[resolve_loader\(\)](#) ([moderngl_window.resources.textures.Textures](#)
[method](#)), 99
[resolved_path](#) ([moderngl_window.meta.base.ResourceDescription](#)
[attribute](#)), 79
[resolved_path](#) ([moderngl_window.meta.data.DataDescription](#)
[attribute](#)), 87
[resolved_path](#) ([moderngl_window.meta.program.ProgramDescription](#)
[attribute](#)), 84
[resolved_path](#) ([moderngl_window.meta.scene.SceneDescription](#)
[attribute](#)), 85
[resolved_path](#) ([moderngl_window.meta.texture.TextureDescription](#)
[attribute](#)), 81
[resource_dir](#) ([moderngl_window.context.base.window.WindowConfig](#)
[attribute](#)), 28
[resource_type](#) ([moderngl_window.meta.base.ResourceDescription](#)
[attribute](#)), 80
[resource_type](#) ([moderngl_window.meta.data.DataDescription](#)
[attribute](#)), 87
[resource_type](#) ([moderngl_window.meta.program.ProgramDescription](#)
[attribute](#)), 84
[resource_type](#) ([moderngl_window.meta.scene.SceneDescription](#)
[attribute](#)), 86
[resource_type](#) ([moderngl_window.meta.texture.TextureDescription](#)
[attribute](#)), 82
[ResourceDescription](#) (in module [moderngl_window.meta.base](#)), 79
[rotate\(\)](#) ([moderngl_window.scene.KeyboardCamera](#)
[method](#)), 109
[window_config\(\)](#) (in module [moderngl_window](#)), 16

S

[sampler](#) ([moderngl_window.scene.MaterialTexture](#)
[attribute](#)), 114
[samples](#) ([moderngl_window.context.base.window.BaseWindow](#)
[attribute](#)), 31
[samples](#) ([moderngl_window.context.base.window.WindowConfig](#)
[attribute](#)), 28
[samples](#) ([moderngl_window.context.glfw.window.Window](#)
[attribute](#)), 36
[samples](#) ([moderngl_window.context.headless.window.Window](#)
[attribute](#)), 40
[samples](#) ([moderngl_window.context.pyglet.window.Window](#)
[attribute](#)), 46
[samples](#) ([moderngl_window.context.pyqt5.window.Window](#)
[attribute](#)), 51
[samples](#) ([moderngl_window.context.pyside2.window.Window](#)
[attribute](#)), 56
[samples](#) ([moderngl_window.context.sdl2.window.Window](#)
[attribute](#)), 60
[SCENE_DIRS](#) ([moderngl_window.conf.Settings](#)
[attribute](#)), 20
[SCENE_FINDERS](#) ([moderngl_window.conf.Settings](#)
[attribute](#)), 20
[SCENE_LOADERS](#) ([moderngl_window.conf.Settings](#)
[attribute](#)), 20
[SceneDescription](#) (in module [moderngl_window.meta.scene](#)), 84
[Scenes](#) (in module [moderngl_window.resources.scenes](#)), 100
[SCREENSHOT_PATH](#) ([moderngl_window.conf.Settings](#)
[attribute](#)), 19

<code>set_default_viewport()</code> (moderngl_window.context.base.window.BaseWindow method), 29	<code>settings_attr</code> (moderngl_window.resources.textures.Textures attribute), 99
<code>set_default_viewport()</code> (moderngl_window.context.glfw.window.Window method), 33	<code>setup_basic_logging()</code> (in module moderngl_window), 15
<code>set_default_viewport()</code> (moderngl_window.context.headless.window.Window method), 39	<code>size</code> (moderngl_window.context.base.window.BaseWindow attribute), 30
<code>set_default_viewport()</code> (moderngl_window.context.pyglet.window.Window method), 43	<code>size</code> (moderngl_window.context.glfw.window.Window attribute), 35
<code>set_default_viewport()</code> (moderngl_window.context.pyqt5.window.Window method), 48	<code>size</code> (moderngl_window.context.headless.window.Window attribute), 39
<code>set_default_viewport()</code> (moderngl_window.context.pyside2.window.Window method), 53	<code>size</code> (moderngl_window.context.pyglet.window.Window attribute), 45
<code>set_default_viewport()</code> (moderngl_window.context.sdl2.window.Window method), 58	<code>size</code> (moderngl_window.context.pyqt5.window.Window attribute), 50
<code>set_position()</code> (moderngl_window.scene.Camera method), 107	<code>size</code> (moderngl_window.context.pyside2.window.Window attribute), 55
<code>set_position()</code> (moderngl_window.scene.KeyboardCamera method), 109	<code>size</code> (moderngl_window.context.sdl2.window.Window attribute), 59
<code>Settings</code> (in module moderngl_window.conf), 17	<code>sphere()</code> (in module moderngl_window.geometry), 64
<code>settings_attr</code> (moderngl_window.finders.base.BaseFilesystemFinder attribute), 89	<code>start()</code> (moderngl_window.timers.base.BaseTimer method), 103
<code>settings_attr</code> (moderngl_window.finders.data.FilesystemFinder attribute), 91	<code>start()</code> (moderngl_window.timers.clock.Timer method), 104
<code>settings_attr</code> (moderngl_window.finders.program.FilesystemFinder attribute), 90	<code>stop()</code> (moderngl_window.timers.base.BaseTimer method), 103
<code>settings_attr</code> (moderngl_window.finders.scene.FilesystemFinder attribute), 90	<code>stop()</code> (moderngl_window.timers.clock.Timer method), 104
<code>settings_attr</code> (moderngl_window.finders.texture.FilesystemFinder attribute), 90	<code>supported_extensions</code> (moderngl_window.loaders.scene.gltf2.Loader attribute), 73
<code>settings_attr</code> (moderngl_window.resources.base.BaseRegistry attribute), 98	<code>supports_file()</code> (moderngl_window.loaders.base.BaseLoader class method), 65
<code>settings_attr</code> (moderngl_window.resources.data.DataFiles attribute), 101	<code>supports_file()</code> (moderngl_window.loaders.data.binary.Loader class method), 77
<code>settings_attr</code> (moderngl_window.resources.programs.Programs attribute), 100	<code>supports_file()</code> (moderngl_window.loaders.data.json.Loader class method), 75
<code>settings_attr</code> (moderngl_window.resources.scenes.Scenes attribute), 101	<code>supports_file()</code> (moderngl_window.loaders.data.text.Loader class method), 76
	<code>supports_file()</code> (moderngl_window.loaders.program.separate.Loader class method), 69
	<code>supports_file()</code> (moderngl_window.loaders.program.single.Loader class method), 67
	<code>supports_file()</code> (moderngl_window.loaders.scene.gltf2.Loader class method), 72
	<code>supports_file()</code> (moderngl_window.loaders.scene.stl.Loader class method), 72

[method](#)), 74
[supports_file\(\)](#) ([moderngl_window.loaders.scene.wavefront.Loader](#) class method), 71
[supports_file\(\)](#) ([moderngl_window.loaders.texture.array.Loader](#) class method), 70
[supports_file\(\)](#) ([moderngl_window.loaders.texture.t2d.Loader](#) class method), 66
[swap_buffers\(\)](#) ([moderngl_window.context.base.window.BaseWindow](#) method), 29
[swap_buffers\(\)](#) ([moderngl_window.context.glfw.window.Window](#) method), 33
[swap_buffers\(\)](#) ([moderngl_window.context.headless.window.Window](#) method), 38
[swap_buffers\(\)](#) ([moderngl_window.context.pyglet.window.Window](#) method), 43
[swap_buffers\(\)](#) ([moderngl_window.context.pyqt5.window.Window](#) method), 48
[swap_buffers\(\)](#) ([moderngl_window.context.pyside2.window.Window](#) method), 53
[swap_buffers\(\)](#) ([moderngl_window.context.sdl2.window.Window](#) method), 58
[time](#) ([moderngl_window.timers.clock.Timer](#) attribute), 104
[Timer](#) (in module [moderngl_window.timers.clock](#)), 104
[title](#) ([moderngl_window.context.base.window.BaseWindow](#) attribute), 30
[title](#) ([moderngl_window.context.base.window.WindowConfig](#) attribute), 27
[title](#) ([moderngl_window.context.glfw.window.Window](#) attribute), 35
[title](#) ([moderngl_window.context.headless.window.Window](#) attribute), 39
[title](#) ([moderngl_window.context.pyglet.window.Window](#) attribute), 45
[title](#) ([moderngl_window.context.pyqt5.window.Window](#) attribute), 50
[title](#) ([moderngl_window.context.pyside2.window.Window](#) attribute), 55
[title](#) ([moderngl_window.context.sdl2.window.Window](#) attribute), 59
[to_dict\(\)](#) ([moderngl_window.conf.Settings](#) method), 18
[tobytes\(\)](#) ([moderngl_window.opengl.projection.Projection3D](#) method), 93
[toggle_pause\(\)](#) ([moderngl_window.timers.base.BaseTimer](#) method), 103
[toggle_pause\(\)](#) ([moderngl_window.timers.clock.Timer](#) method), 104
[transform\(\)](#) ([moderngl_window.opengl.vao.VAO](#) method), 95

T

[tess_control_shader](#) ([moderngl_window.meta.program.ProgramDescription](#) attribute), 83
[tess_evaluation_shader](#) ([moderngl_window.meta.program.ProgramDescription](#) attribute), 83
[texture](#) ([moderngl_window.scene.MaterialTexture](#) attribute), 114
[TEXTURE_DIRS](#) ([moderngl_window.conf.Settings](#) attribute), 20
[TEXTURE_FINDERS](#) ([moderngl_window.conf.Settings](#) attribute), 20
[TEXTURE_LOADERS](#) ([moderngl_window.conf.Settings](#) attribute), 20
[TextureDescription](#) (in module [moderngl_window.meta.texture](#)), 80
[Textures](#) (in module [moderngl_window.resources.textures](#)), 98
[time](#) ([moderngl_window.timers.base.BaseTimer](#) attribute), 104

U

[unicode_char_entered\(\)](#) ([moderngl_window.context.base.window.WindowConfig](#) method), 25
[unicode_char_entered_func](#) ([moderngl_window.context.base.window.BaseWindow](#) attribute), 32
[unicode_char_entered_func](#) ([moderngl_window.context.glfw.window.Window](#) attribute), 37
[unicode_char_entered_func](#) ([moderngl_window.context.headless.window.Window](#) attribute), 41
[unicode_char_entered_func](#) ([moderngl_window.context.pyglet.window.Window](#) attribute), 46
[unicode_char_entered_func](#) ([moderngl_window.context.pyqt5.window.Window](#) attribute), 51
[unicode_char_entered_func](#) ([moderngl_window.context.pyside2.window.Window](#) attribute), 56

[unicode_char_entered_func](#) (moderngl_window.context.sdl2.window.Window attribute), 61

[update\(\)](#) (moderngl_window.opengl.projection.Projection3D method), 93

[use\(\)](#) (moderngl_window.context.base.window.BaseWindow method), 29

[use\(\)](#) (moderngl_window.context.glfw.window.Window method), 33

[use\(\)](#) (moderngl_window.context.headless.window.Window method), 38

[use\(\)](#) (moderngl_window.context.pyglet.window.Window method), 42

[use\(\)](#) (moderngl_window.context.pyqt5.window.Window method), 48

[use\(\)](#) (moderngl_window.context.pyside2.window.Window method), 53

[use\(\)](#) (moderngl_window.context.sdl2.window.Window method), 58

W

[width](#) (moderngl_window.context.base.window.BaseWindow attribute), 30

[width](#) (moderngl_window.context.glfw.window.Window attribute), 35

[width](#) (moderngl_window.context.headless.window.Window attribute), 39

[width](#) (moderngl_window.context.pyglet.window.Window attribute), 45

[width](#) (moderngl_window.context.pyqt5.window.Window attribute), 50

[width](#) (moderngl_window.context.pyside2.window.Window attribute), 55

[width](#) (moderngl_window.context.sdl2.window.Window attribute), 59

[WINDOW](#) (moderngl_window.conf.Settings attribute), 19

[window\(\)](#) (in module moderngl_window), 15

[window_size](#) (moderngl_window.context.base.window.WindowConfig attribute), 27

[WindowConfig](#) (in module moderngl_window.context.base.window), 23

V

[VAO](#) (in module moderngl_window.opengl.vao), 94

[vertex_shader](#) (moderngl_window.meta.program.ProgramDescription attribute), 83

[viewport](#) (moderngl_window.context.base.window.BaseWindow attribute), 30

[viewport](#) (moderngl_window.context.glfw.window.Window attribute), 35

[viewport](#) (moderngl_window.context.headless.window.Window attribute), 40

[viewport](#) (moderngl_window.context.pyglet.window.Window attribute), 45

[viewport](#) (moderngl_window.context.pyqt5.window.Window attribute), 50

[viewport](#) (moderngl_window.context.pyside2.window.Window attribute), 55

[viewport](#) (moderngl_window.context.sdl2.window.Window attribute), 59

[vsync](#) (moderngl_window.context.base.window.BaseWindow attribute), 30

[vsync](#) (moderngl_window.context.glfw.window.Window attribute), 36

[vsync](#) (moderngl_window.context.headless.window.Window attribute), 40

[vsync](#) (moderngl_window.context.pyglet.window.Window attribute), 45

[vsync](#) (moderngl_window.context.pyqt5.window.Window attribute), 50

[vsync](#) (moderngl_window.context.pyside2.window.Window attribute), 55

[vsync](#) (moderngl_window.context.sdl2.window.Window attribute), 60