

---

# **bgdev**

***Release 1.0.0***

**unknown**

**Dec 10, 2020**



**GENERAL**

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Installation</b>	<b>5</b>
<b>3</b>	<b>Usage</b>	<b>7</b>
<b>4</b>	<b>bgdev</b>	<b>9</b>
<b>5</b>	<b>Release Notes</b>	<b>49</b>
<b>6</b>	<b>Module Index</b>	<b>51</b>
<b>7</b>	<b>Index</b>	<b>53</b>
	<b>Python Module Index</b>	<b>55</b>
	<b>Index</b>	<b>57</b>



Tools and libraries for Character Riggers and TDs (mostly for Autodesk Maya)



## INTRODUCTION

bgdev is a custom suite of python tools and utility function, mostly created for Autodesk Maya and focusing on TD roles like Character Rigging or Animation. It started with my career back in August 2013, and has grown over time with the experience I got from various jobs in different companies.

### 1.1 License

The current repository is under [MIT License](#).

Feel free to use, change, and share it as you please. You don't have to, but mentioning my name whenever you use source code from here would be much appreciated!

### 1.2 API Documentation

You can find a generated sphinx documentation at <https://bgdev.readthedocs.io/en/latest/>





## **INSTALLATION**

bgdev doesn't require anything apart from a working version of Autodesk Maya. You will find a module file available in `bgdev\src\apps-maya\module\modules\` which you can add to the `MAYA_MODULE_PATH` environment variable. It'll allow Maya to pick up the whole repo automatically for you on startup.

You can always run `sys.path.append()` on the various Python source folders: `bgdev\src\apps-maya\python` and `bgdev\src\apps-python`. Don't forget to include the Python external libraries in `bgdev\src\site-packages\Python27` too or some tools may not be working!



---

## CHAPTER THREE

---

### USAGE

Once the module is installed, all you need to do is to run `import bgdev` inside Maya. *(Obviously you need to call the right tool/library, but if you found this repo you probably know how python works! :-))*



## 4.1 bgdev.tools

### 4.1.1 bgdev.tools.mayaColorPicker

#### bgdev.tools.mayaColorPicker.core

Color picker for Maya. (old)

**created** 25/02/2017

**author** Benoit GIELLY <benoit.gielly@gmail.com>

#### bgdev.tools.mayaColorPicker.main

Maya Color Picker

**created** 20/02/2018

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**class** **ColorPicker** (*name=None, index=17*)

Bases: object

**\_\_init\_\_** (*name=None, index=17*)

Initialize self. See help(type(self)) for accurate signature.

**property** **color**

**property** **node**

### 4.1.2 bgdev.tools.pluginManager

#### bgdev.tools.pluginManager.core

Custom plugin manager for Maya.

**created** 10/07/2017

**author** Benoit GIELLY <benoit.gielly@gmail.com>

### 4.1.3 bgdev.tools.renamer

Convenient wrapper to the core and ui modules.

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**show** (*\*args, \*\*kwargs*)

Wrap the show function of the core module.

#### bgdev.tools.renamer.core

Renamer tool for maya.

**created** 06/04/2016

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**show** (*parent=None*)

Show widget.

#### bgdev.tools.renamer.ui

Main UI for the renamer.

**created** 06/04/2016

**author** Benoit Gielly <benoit.gielly@gmail.com>

**maya\_window** ()

Return the MayaWindow if in Maya else None.

#### bgdev.tools.renamer.utils

Utils function for the renamer.

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**UNDO** (*func, \*, undo=True*)

Decorate a method to make it undoable.

**check\_image** (*icon, normalized=False, as\_icon=False*)

Convert the given image path to full path name.

##### Parameters

- **icon** (*str*) – Name of the icon to check
- **normalized** (*bool*) – Normalize the path to slash forward (works on all OS)
- **as\_icon** (*bool*) – Return a QIcon of the found image file.

**Returns** Updated icon path so it's ready to be used.

**Return type** str

**method\_decorator** (*func, undo=False*)

Decorate a method to make it undoable.

#### 4.1.4 bgdev.tools.toolbar

##### bgdev.tools.toolbar.tabs

##### bgdev.tools.toolbar.tabs.maya

##### bgdev.tools.toolbar.tabs.maya.main

Maya toolbar override.

**created** 14/10/2018

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**dock\_widget** (*widget*, *title*='DockWindow', *area*='right', *floating*=False)  
Dock the given widget properly for both M2016 and 2017+.

**get\_workspaces** (*\*\*kwargs*)  
Get Maya workspaces (WIP).

**maya\_window** ()  
Return the MayaWindow if in Maya else None.

**show** (*\*args*, *\*\*kwargs*)  
Show the widget within a maya workspaceControl.

**to\_qwidget** (*ctrl*)  
Convert a Maya widget to a PySide2 QWidget.

**Parameters** **ctrl** (*str*) – Name of the maya widget as a string.

**Returns** QWidget instance object of the given widget.

**Return type** QtWidgets.QWidget

##### bgdev.tools.toolbar.tabs.system

##### bgdev.tools.toolbar.tabs.system.main

Maya toolbar override.

**created** 14/10/2018

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**show** ()  
Show the window.

**bgdev.tools.toolbar.cfg**

Config file to be shared in the toolbar project.

**created** 26/11/2020

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**bgdev.tools.toolbar.main**

Main toolbar code.

**created** 09/11/2018

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**get\_class\_name** (*cls*)

Get the class full path name.

**show** (*\*args, \*\*kwargs*)

Show the window.

**bgdev.tools.toolbar.ui**

PySide Widgets used in the main toolbar code.

**created** 09/11/2018

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**bgdev.tools.toolbar.utils**

Utility method used in the toolbar.

**created** 12/11/2018

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**check\_image** (*icon, normalized=True*)

Convert the given image path to full path name.

**Parameters**

- **icon** (*str*) – Name of the icon to check
- **normalized** (*bool*) – Normalize the path to work on all OS

**Returns** Updated icon path so it's ready to be used.

**Return type** *str*

**check\_modpath\_has\_init** (*path, parts*)

Check if given module parts all contains an `__init__` file.

**Parameters**

- **path** (*str*) – Path to python file.
- **parts** (*list*) – Each parts of a module path. (Eg. ["sanity", "core"]).

**Returns** True if each part as an `__init__` file as sibling else False.

**Return type** *bool*



**create\_module\_callback** (*source*)

Set the command for each buttons.

**Parameters** **source** (*str*) – Source python code to execute.

**Returns** A callable function

**Return type** function

**get\_dcc** ()

Get current DCC application.

**get\_dcc\_package** ()

Get the current DCC's "check" package.

**modpath\_from\_file** (*filename*)

Get the module import path from its filename.

**Parameters** **filename** (*str*) – Path to python file.

**Raises** **ImportError** – When module cannot be found in sys.path.

**Returns** The module import path.

**Return type** str

**yaml\_dump** (*data, path, ordered=False, \*\*kwargs*)

Export YAML file.

**Parameters**

- **data** (*dict*) – Dictionary to export as JSON
- **path** (*str*) – YAML file path to save data
- **ordered** (*bool*) – Dump data ordered when True.
- **kwargs** (*dict*) – Any extra flags to pass in the `json.dump()` command

**Returns** given file path

**Return type** str

**yaml\_load** (*path, ordered=False, \*\*kwargs*)

Import YAML file.

**Parameters**

- **path** (*str*) – YAML file path to save data
- **ordered** (*bool*) – Load data in an OrderedDict when True.
- **kwargs** (*dict*) – Any extra flags to pass in the `json.load()` command

**Returns** the loaded data

**Return type** dict

### 4.1.5 bgdev.tools.atom\_import\_export

Use ATOM to import or export animation.

**created** 31/01/2019

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**atom\_export** (*path, nodes*)

Export selected controllers into atom file.

**Parameters**

- **path** (*str*) – The atom file path.
- **nodes** (*list*) – List of controllers to export animation from.

**atom\_import** (*path, nodes*)

Import animation curves from atom file.

**Parameters**

- **path** (*str*) – The atom file path.
- **nodes** (*list*) – List of controllers to import animation onto.

**export\_dialog** ()

Open up a file dialog to get the atom file to export.

**import\_dialog** ()

Open up a file dialog to get the atom file to import.

### 4.1.6 bgdev.tools.bindpose

Create and set bindpose on controllers.

**created** 17/01/2019

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**create\_bindpose** ()

Create a bindpose attribute on every controls.

**get\_all\_controls** ()

Get all rig controllers in the scene.

**restore\_bindpose** (*selected=True*)

Restore the rig controls to their bindpose.

**Parameters** **selected** (*bool*) – If True, reset only selected controls, otherwise reset every controls of each selected rigs.

### 4.1.7 bgdev.tools.curve

Custom tools with curves.

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**attach\_nodes\_to\_curve** ()

Attach selected nodes to selected curve.

**curve\_from\_nodes** (*degree=2*)

Create a curve and snap each CVs on each selected nodes.

**Parameters** **degree** (*int*) – The curve degree.

**Returns** The created curve node.

**Return type** str

**get\_closest\_point** (*point, node\_list*)

Get the closest point to each nodes in the list.

**Parameters**

- **point** (*str*) – Node to evaluate the distance to.
- **node\_list** (*list*) – List of nodes to search for the closest to the point.

**Returns** The closest node.

**Return type** str

### 4.1.8 bgdev.tools.mel2py

Convert mel to python.

**created** 23/01/2019

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**convert\_command** (*command, namespace='cmds', verbose=True*)

Convert mel command-string to python.

**Parameters**

- **command** (*str*) – The mel command as a string.
- **namespace** (*str*) – The namespace to use for the module.
- **verbose** (*bool*) – Prints the command after conversion.

**Returns** The convert mel command to python.

**Return type** str

**convert\_file** (*files, force\_compatibility=True*)

Convert mel files to python.

**Parameters**

- **files** (*list*) – List of path to mel files.
- **force\_compatibility** (*bool*) – Pymel argument. Don't really know if it has an impact or not so adding it as a kwarg just in case.

### 4.1.9 bgdev.tools.outliner\_color

Tool to quickly update outliner color of selected nodes.

**created** 11/01/2019

**author** Benoit GIELLY <benoit.gielly@gmail.com>

### 4.1.10 bgdev.tools.performance

Evaluation Toolkit.

**created** 03/06/2019

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**class Profiler**

Bases: object

Create a python profiler to check for code usage.

#### Example

```
import bgdev.tools.performance
profiler = bgdev.tools.performance.Profiler()
profiler.start()
# run python code
profiler.stop()
```

**\_\_init\_\_()**

Initialize self. See help(type(self)) for accurate signature.

**start()**

Enable the profiler.

**stats (sort='tottime')**

Print the profiler stats.

**Parameters sort (str)** – Sort the stats with given value.

**stop (sort='time')**

Stop the profiler and print result.

**Parameters sort (str)** – Sort the stats with given value.

**frames\_per\_second (iterations=3, start=1001, frames=100, viewports='viewport2', meshes\_only=True, outfile=None)**

Evaluate the framePerSecond in the current scene.

#### Parameters

- **iterations (int)** – Amount of time the timeline should run.
- **start (int)** – Start frame.
- **frames (int)** – The amount of frames to run.
- **viewports (list)** – Select the viewports you want to evaluate on. Possible values are “legacy” and “viewport2”.
- **meshes\_only (bool)** – Hides everything but meshes in the viewport.

- **outfile** (*bool or str*) – Save the output fps in the given txt file. If *True* is passed, query the current scene path and save the log next to it (Useful when benchmarking multiple scenes in a loop).

**Returns** The evaluated frame per seconds as a float number.

**Return type** float

#### 4.1.11 bgdev.tools.pypredef

Generate pypredefs using Maya’s help query to read command flags.

**created** 18/05/2018

**author** Benoit GIELLY <[benoit.gielly@gmail.com](mailto:benoit.gielly@gmail.com)>

**cleanup\_flags** (*docstring*)

Cleanup the docstring from the mel output to a nicer result.

**Parameters** **docstring** (*str*) – The docstring extracted from the help command query.

##### Example

Reformat this docstring:

```
Flags:
  -n -name          String
  -p -parent        String
  -s -shared
  -ss -skipSelect
```

So it looks like this:

```
Flags:
  name (n): String
  parent (p): String
  shared (s): None
  skipSelect (ss): None
```

Also returns a list of all the flags so they can be added to the method definition (eg. “flag=None”) so IDE can autocomplete them.

**Parameters** **docstring** (*str*) – The docstring containing the flags to cleanup.

**Returns** The cleaned-up docstring with nicer flags, and the list of flags

**Return type** tuple

**generate\_pypredef** (*module, outdir=None, doclink=True*)

Generate pypredefs using Maya’s help query to read command flags.

**Parameters**

- **module** (*mod*) – Module from which you want to generate the pypredef.
- **outdir** (*str*) – Directory in which you want to export the text file.
- **doclink** (*bool*) – Adds a link to the online documentation.

---

**Note:** There are a few maya commands (e.g. “nexCtx”) that are almost 1 MILLION line length... so there’s a test to skip the command if the docstring is longer than 10k lines.

---

### Example

```
from maya import cmds
generate_pypredef(cmds, outdir="~/Desktop")
```

#### **get\_maya\_urls()**

Generate a table for each commands from the online documentation.

**Returns** Generates a {command:url} dictionary.

**Return type** dict

#### **indent(text, amount=4)**

Indent each lines of the given string.

##### **Parameters**

- **text** (*str*) – Text to indent
- **amount** (*int*) – Number or character padding

**Returns** Indented text.

**Return type** str

#### **iskeyword()**

x.\_\_contains\_\_(y) <==> y in x.

## 4.1.12 bgdev.tools.symmetry

Utility methods used to sort nodes and graphs.

**created** 20/11/2020

**author** Benoit GIELLY <benoit.gielly@gmail.com>

#### **class Symmetry**

Bases: object

Generates a symmetry table for selected mesh.

##### **\_\_init\_\_()**

Initialize self. See help(type(self)) for accurate signature.

##### **property center**

Get center vertices.

##### **get\_side\_vertices(side)**

Get all components of given side.

##### **static get\_vertex\_id(vertex)**

Get vertex index from name.

##### **get\_vertex\_name(index)**

Get vertex name from index.

**property left**

Get left side vertices.

**mirror** (*vertices*)

Mirror vertices.

**Parameters** *vertices* (*list*) – List of vertices to mirror.

**Yields** *str* – The next vertex full name in the given list.

**mirror\_selection** (*add=False*)

Mirror selected vertices.

**Parameters** *add* (*bool*) – Add to existing selection when True.

**populate\_table** ()

Populate the symmetry table with vertices.

**Notes**

Maya topology symmetry must be already activated.

**property right**

Get right side vertices.

**update** ()

Generate a symmetry table.

### 4.1.13 bgdev.tools.symmetry\_api

Utility methods used to sort nodes and graphs.

**created** 20/11/2020

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**class Symmetry**

Bases: `object`

Generates a symmetry table for selected mesh.

**\_\_init\_\_** ()

Initialize self. See help(type(self)) for accurate signature.

**property center**

Get center vertices.

**get\_side\_vertices** (*side*)

Get all components of given side.

**Parameters** *side* (*str*) – Side of the table to query. Either “left”, “center” or “right”.

**Returns** List of side-vertices.

**Return type** list

**static get\_sides\_from\_rich** (*vertices*)

Get resulting sides of a RichSelection in symmetry.

**Parameters** *vertices* (*list*) – Vertices to select for symmetry.

**Returns**

Two *OpenMaya.MSelectionList*, being the result of the `MRichSelection.getSelection` and `MRichSelection.getSymmetry` methods.

**Return type** tuple

**static** `get_vertex_id` (*vertex*)

Get vertex index from name.

**Parameters** `vertex` (*str*) – Full vertex name (eg. “mesh.vtx[15]”)

**Returns** Index of the vertex as integer.

**Return type** int

`get_vertex_name` (*index*)

Get vertex name from index.

**Parameters** `index` (*int*) – Index of the vertex.

**Returns** Full vertex name (eg. “mesh.vtx[15]”)

**Return type** str

**property** `left`

Get left side vertices.

**mirror** (*vertices*)

Mirror vertices.

**Parameters** `vertices` (*list*) – List of vertices to mirror.

**Yields** *str* – The next vertex full name in the given list.

**mirror\_selection** (*add=False*)

Mirror selected vertices.

**Parameters** `add` (*bool*) – Add to existing selection when True.

**populate\_table** ()

Populate the symmetry table with vertices.

## Notes

Maya topology symmetry must be activated.

**property** `right`

Get right side vertices.

**update** ()

Generate a symmetry table.

### 4.1.14 bgdev.tools.uniloop

Create an easy attribute setting loop.

**created** 14/10/2016

**author** Benoit GIELLY <benoit.gielly@gmail.com>



### 4.1.15 bgdev.tools.vector\_plane

Symmetry stuff.

**created** 14/01/2019

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**get\_mirrorvector** (*obj*, *\*\*kwargs*)

**mirrorToolCMD** (*\*args*, *\*\*kwargs*)

**mirror\_layout** ()

**setMirrorToolCMD** (*callback*, *fields=None*)

**vector** (*\*args*)

## 4.2 bgdev.ui

### 4.2.1 bgdev.ui.dock

Make any window dockable within Maya.

**created** 8 Jun 2018

**author** Benoit Gielly <benoit.gielly@gmail.com>

**dock\_widget** (*widget*, *label='DockWindow'*, *area='right'*, *floating=False*)

Dock the given widget properly for both M2016 and 2017+.

**get\_available\_controls** ()

Return all the available Controls in list.

**Returns** List of existing workspaceControls.

**Return type** list

### 4.2.2 bgdev.ui.flow\_layout

Custom FlowLayout.

**author** Benoit GIELLY <benoit.gielly@gmail.com>

### 4.2.3 bgdev.ui.progress\_bar

ProgressBar class to use in Maya.

---

**Todo:** Convert to PySide and move to `bgdev.ui.widget`

---

**created** 23/05/2018

**author** Benoit Gielly <benoit.gielly@gmail.com>

**class ProgressBar**

Bases: `object`

ProgressBar class.

```
__init__ ()  
    Constructor.  
  
end ()  
    Finish the process.  
  
is_cancelled ()  
    Check if progress was cancelled (esc. key pressed).  
  
start (status='Begin operation...', value=100)  
    Start the process.  
  
property status  
    Update the status of the progressBar.  
  
    Type str  
  
step (step=1)  
    Increase progress step by one.  
  
class ProgressBar2  
    Bases: object  
    ProgressBar class.  
  
    __init__ ()  
        Constructor.  
  
    end ()  
        Finish the process.  
  
    start (status='Begin operation...', maximum=100)  
        Start the process.  
  
    property status  
        Update the status of the progressBar.  
  
        Type str  
  
    step ()  
        Increase progress step by one.
```

## 4.2.4 bgdev.ui.utils

Utility fonctions for UI.

**created** 8/02/2018

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**check\_image** (icon, normalized=True, as\_icon=False)  
Convert the given image path to full path name.

### Parameters

- **icon** (str) – Name of the icon to check
- **normalized** (bool) – Normalize the path to slash forward (works on all OS)
- **as\_icon** (bool) – Return a QIcon of the found image file.

**Returns** Updated icon path so it's ready to be used.

**Return type** str

**generate\_qrc** (*path*, *name*='resources', *prefix*='icons')

Generate .qrc file based on given folder.

**Parameters**

- **path** (*str*) – Path to icons folder.
- **name** (*str*) – Name of the qrc file to generate. Default is “resources”.
- **prefix** (*str*) – Name of the icon prefix. Default is “icons”.

**Returns** Path to the generated [qrc].py file.

**Return type** str

**in\_maya** ()

Check if we are in Maya or not.

**Returns** True if in maya else False

**Return type** bool

**main\_window** ()

Return the QMainWindow for the current application.

**maya\_window** ()

Get Maya MainWindow as Qt.

**Returns** Maya main window as QObject

**Return type** QtWidgets.QWidget

**to\_qwidget** (*ctrl*)

Convert a Maya widget to a PySide2 QWidget.

**Parameters** **ctrl** (*str*) – Name of the maya widget as a string.

**Returns** QWidget instance object of the given widget.

**Return type** QtWidgets.QWidget

## 4.2.5 bgdev.ui.widgets

PySide custom widgets.

**created** 12/12/2018

**author** Benoit GIELLY <benoit.gielly@gmail.com>

## 4.3 bgdev.utils

### 4.3.1 bgdev.utils.attribute

**created** 13 Feb 2018

**author** Benoit Gielly <benoit.gielly@gmail.com>

**attributes\_restore** (*node*)

Restore previously unlocked attributes to their default state.

**Parameters** **node** (*str*) – Node to restore attributes

**Returns** False if attribute doesn't exists else True

**Return type** bool

**attributes\_toggle** (*nodes=None, attr\_list=None*)

Toggle attributes between default and unlocked states.

**Parameters**

- **nodes** (*list*) – List of nodes to toggle. If not given, use viewport selection.
- **attr\_list** (*list*) – List of extra attributes to toggle

**attributes\_unlock** (*node, base=True, user=True, attr\_list=None*)

Unlock attributes on given node

**Parameters**

- **node** (*str*) – Node to unlock attributes
- **attr\_list** (*list*) – List of extra attributes to toggle
- **base** (*bool*) – If True, will toggle base attributes (translate, rotate, scale, visibility).
- **user** (*bool*) – If True, will toggle user's attributes.

**create\_separator\_attribute** (*node, name=None*)

Create a separator attribute on the given node.

**Parameters**

- **node** (*str*) – Name of the node to add the separator attribute.
- **name** (*str*) – Name nice (category)

## 4.3.2 bgdev.utils.bindpose

Create and set bindpose on controllers.

**created** 17/01/2019

**author** Benoit GIELLY <[benoit.gielly@gmail.com](mailto:benoit.gielly@gmail.com)>

**create\_bindpose** ()

Create a bindpose attribute on every controls.

**restore\_bindpose** (*selected=True*)

Restore the rig controls to their bindpose.

**Parameters** **selected** (*bool*) – If True, reset only selected controls, otherwise reset every controls of each selected rigs.

## 4.3.3 bgdev.utils.color

Color utilites.

**created** 08/02/2019

**author** Benoit GIELLY <[benoit.gielly@gmail.com](mailto:benoit.gielly@gmail.com)>

**convert\_css\_color** (*css\_color*)

Convert given color using the PySide2 CSS API.

**Parameters** **css\_color** (*str*) – Any color-string you would type in a Widget's CSS.

**Returns** The RGB-converted color.

**Return type** tuple

### 4.3.4 bgdev.utils.connections

**created** 2017-03-10

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**pairblend\_three\_nodes** ()

Create pairblend between three nodes.

**quick\_connections** ()

Quickly connect transforms.

### 4.3.5 bgdev.utils.constraints

Utility methods about constraints.

**created** 05/03/2017

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**aim\_to\_children** (*up\_vector*=(0, 1, 0), *aim\_vector*=(1, 0, 0))

Find an up vector for the selection and aim toward each others.

**Parameters**

- **up\_vector** –
- **aim\_vector** –

**matrix\_constraint** (*driver*, *target*, *srt*='srt')

Constraint one node to another using their worldMatrix attributes.

**Parameters**

- **driver** (*str*) – The driver node
- **target** (*str*) – The node that follow the driver.
- **srt** (*str*) – The attributes to connect.

**Returns** The multMatrix and decomposeMatrix nodes.

**Return type** tuple

**matrix\_constraint\_callback** (*srt*='trs')

Call back update `intermediate()`.

### 4.3.6 bgdev.utils.controller

Utilities for controllers.

Mostly, controller shapes creation (needs to be turned into data)

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**create\_control** (*ctrl*, *variant*=0, *name*=None, *size*=1.0, *normal*=(0, 1, 0), *color*='yellow')

Create a controller based on its name and variant.

**Parameters**

- **ctrl** (*str*) – The type of controller you want to create. Use `list_controllers()` method to know which ones are available.
- **variant** (*int*) – Which variant of the controller type you want.
- **name** (*str*) – Rename the controller with the given name.
- **size** (*float*) – Scale the controller using given size value.
- **normal** (*tuple*) – Which axis should the controller aim to when created. Default axis is +Y
- **color** (*str*) – The name of the CSS color to apply as an RGB override. Uses the PySide2 StyleSheet API to convert names to RGB color.

**Returns** The controller and its shape.

**Return type** tuple

**get\_all\_controls** (*namespace=""*)

Get all rig controllers in the scene.

**get\_ctrl\_data** (*name, variant=0*)

Query the controller data from its name.

**get\_rig\_controls** ()

Query all controllers of the selected rig(s).

**list\_controllers** ()

List all the available controls.

**select\_all** ()

Select all controllers in the current scene.

**select\_rig\_controls** ()

Select all controllers from selected rig(s).

### 4.3.7 bgdev.utils.curve

Utility methods for curves.

**created** 04/02/2019

**author** Benoit Gielly <[benoit.gielly@gmail.com](mailto:benoit.gielly@gmail.com)>

**curve\_interpolate** (*curve=None, segments=3*)

Interpolate locators along a selected nurbsCurve.

**Parameters**

- **curve** (*str*) – Name of the curve node. Use selection if None.
- **segments** (*int*) – Amount of segments you want.

**Returns** The created locators.

**Return type** list

### 4.3.8 bgdev.utils.decorator

Useful decorators to use for methods.

**created** 11/12/2018

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**REPEAT** (*func*, \*, *repeat=True*, *undo=False*)

Decorate a method to make it repeatable and/or undoable.

#### Parameters

- **func** (*function*) – The method to decorate.
- **repeat** (*bool*) – Makes the method repeatable.
- **undo** (*bool*) – Makes the method undoable.

#### Example

You can use the custom wrappers in this module to decorate your method as you'd like:

```
import rigging.ui.decorator

@bgdev.ui.decorator.UNDO
def undoable_only_method(*args, **kwargs):
    return args, kwargs

@bgdev.ui.decorator.REPEAT
def repeatable_only_method(*args, **kwargs):
    return args, kwargs

@bgdev.ui.decorator.UNDO_REPEAT
def undoable_repeatable_method(*args, **kwargs):
    return args, kwargs
```

**Returns** The decorated method.

**Return type** function

**UNDO** (*func*, \*, *repeat=False*, *undo=True*)

Decorate a method to make it repeatable and/or undoable.

#### Parameters

- **func** (*function*) – The method to decorate.
- **repeat** (*bool*) – Makes the method repeatable.
- **undo** (*bool*) – Makes the method undoable.

## Example

You can use the custom wrappers in this module to decorate your method as you'd like:

```
import rigging.ui.decorator

@bgdev.ui.decorator.UNDO
def undoable_only_method(*args, **kwargs):
    return args, kwargs

@bgdev.ui.decorator.REPEAT
def repeatable_only_method(*args, **kwargs):
    return args, kwargs

@bgdev.ui.decorator.UNDO_REPEAT
def undoable_repeatable_method(*args, **kwargs):
    return args, kwargs
```

**Returns** The decorated method.

**Return type** function

**UNDO\_REPEAT** (*func*, \*, *repeat=True*, *undo=True*)

Decorate a method to make it repeatable and/or undoable.

**Parameters**

- **func** (*function*) – The method to decorate.
- **repeat** (*bool*) – Makes the method repeatable.
- **undo** (*bool*) – Makes the method undoable.

## Example

You can use the custom wrappers in this module to decorate your method as you'd like:

```
import rigging.ui.decorator

@bgdev.ui.decorator.UNDO
def undoable_only_method(*args, **kwargs):
    return args, kwargs

@bgdev.ui.decorator.REPEAT
def repeatable_only_method(*args, **kwargs):
    return args, kwargs

@bgdev.ui.decorator.UNDO_REPEAT
def undoable_repeatable_method(*args, **kwargs):
    return args, kwargs
```

**Returns** The decorated method.

**Return type** function

**method\_decorator** (*func*, *repeat=False*, *undo=False*)

Decorate a method to make it repeatable and/or undoable.



**Parameters**

- **func** (*function*) – The method to decorate.
- **repeat** (*bool*) – Makes the method repeatable.
- **undo** (*bool*) – Makes the method undoable.

**Example**

You can use the custom wrappers in this module to decorate your method as you'd like:

```
import rigging.ui.decorator

@bgdev.ui.decorator.UNDO
def undoable_only_method(*args, **kwargs):
    return args, kwargs

@bgdev.ui.decorator.REPEAT
def repeatable_only_method(*args, **kwargs):
    return args, kwargs

@bgdev.ui.decorator.UNDO_REPEAT
def undoable_repeatable_method(*args, **kwargs):
    return args, kwargs
```

**Returns** The decorated method.

**Return type** function

**selected** (*func*)

Decorate a func to use selection if no args passed.

**4.3.9 bgdev.utils.deformer**

Utility methods for deformers.

**created** 28/05/2018

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**transfer\_base\_weights** (*source, target, joint=0*)

Transfer weights from source to target.

**Parameters**

- **source** (*str*) – Source deformer
- **target** (*str*) – Target deformer
- **joint** (*int*) – Index of joint in skinCluster

**transfer\_base\_weights\_api** (*source, target, src\_joint=0, tgt\_joint=0*)

Transfer weights from source to target using OpenMaya.

**Parameters**

- **source** (*str*) – Source deformer
- **target** (*str*) – Target deformer

- **joint** (*int*) – Index of joint in skinCluster

**update\_lattice** (*node*, *lattice*, *add=True*)

Add or remove node from lattice.

**Parameters**

- **node** (*str*) – The node to add into the existing lattice.
- **lattice** (*str*) – The lattice (ffd) node.
- **add** (*bool*) – True node to lattice if True, else remove it.

**update\_lattice\_callback** (*add=True*)

Call back `update_lattice()`.

### 4.3.10 bgdev.utils.display

Utility methods about viewport display.

**created** 13/02/2018

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**real\_focus** ()

Zoom to the current selection's manipulator position.

**real\_focus\_old** ()

Zoom to the current selection's manipulator position.

**viewport\_display\_types** (*flag=None*)

Toggle the viewport options.

**Possible flags:**

- allObjects
- nurbsCurves
- polymeshes
- joints
- locators
- etc...

### 4.3.11 bgdev.utils.history

Utility methods to deal with history on nodes.

**created** 11/12/2018

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**delete\_history** (*nodes=None*, *mode=None*)

Delete history on selection.

**Parameters**

- **nodes** (*list*) – List of nodes. If none given, use selection.
- **mode** (*str*) – Can be “pre-deform” or “non-deform”. If None, simply delete all history on selection.

**delete\_unused()**

Delete unused nodes in the scene.

**freeze\_transforms** (*nodes=None, transforms='trsp', history=False*)

Freeze transforms on given nodes.

#### Parameters

- **nodes** (*list*) – List of nodes. If none given, use selection.
- **transforms** (*str*) – Transforms to freeze. e.g. “trs”.
- **history** (*bool*) – Also delete history if True. Default is False.

### 4.3.12 bgdev.utils.hotkey

Easily create custom hotkey.

**created** 11/02/2019

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**create\_hotkey** (*name, command, keys, edit=False, annotation='New user script', category='custom', language='python'*)

Create a user hotkey with its namedCommand.

#### Parameters

- **name** (*str*) – Name of the command to attach to a hotkey
- **command** (*str*) – The command to run. Must be an executable string.
- **keys** (*str*) – The hotkey to use. If you want modifiers, add them before the keyboard key and separate them with “+”. Eg. “alt+shift+A”.
- **edit** (*bool*) – If a command already exists and you want to update it.
- **annotation** (*str*) – Add an annotation to the command/hotkey
- **category** (*str*) – The category in which you add the command. Default is “Custom Scripts.custom”
- **language** (*str*) – The source language used by the command. Must be either “mel” or “python”

**create\_hotkeys\_from\_yaml** (*path=None*)

Quickly re-create custom hotkeys from a yaml file.

**valid\_keyset()**

Make sure the current hotkey set is valid.

### 4.3.13 bgdev.utils.joints

Utility methods about joints.

**created** 13/02/2018

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**add\_joints** (*number=None*)

Add joints inbetween.

**Parameters** **number** (*int*) – amount of joints to add. If none given, a prompt will ask the user.

**Raises**

- **TypeError** – If given type is not a number.
- **RuntimeError** – If no joints are selected, or only one is but has no children.

**show\_joint\_orient** (*state=True*)

Toggle jointOrient and preferredAngle attributes in channelBox.

Applies to all joints in the scene.

**Parameters** **state** (*bool*) – True if you want to display jointOrient in channelbox.

### 4.3.14 bgdev.utils.locators

Utility methods about locators.

**created** 05/03/2017**author** Benoit GIELLY <benoit.gielly@gmail.com>**attach\_locators\_to\_curve** ()

Attach a locator on each curve CVs.

**create\_aim\_locator** (*middle=False*, *aim\_vector=(1, 0, 0)*, *up\_vector=(0, 1, 0)*)

Create a locator that's oriented on the selection.

If selection is 2 or 3, it will use the second as the aim axis, and the third as up. If middle is set to False, it will snap it to the first selected.

**Parameters**

- **middle** (*bool*) – snap in between points 1 and 2 if True, else on point 1.
- **aim\_vector** (*tuple*) – default aim vector for the aimConstraint.
- **up\_vector** (*tuple*) – default up vector for the aimConstraint.

**Returns** The locator created.**Return type** str**Raises** **RuntimeError** – If nothing is selected.**locator\_on\_selection** (*method='matrix'*)

Create and snap a locator on selected nodes.

**Parameters** **method** (*str*) – “matrix” uses Maya’s xform command with matrix flag enabled. “pivot” uses Maya’s xform command with rotatePivot flag enabled. “manip” queries Maya’s move manipulator position and orientation.

### 4.3.15 bgdev.utils.mesh

Utility methods for meshes.

**created** 05/03/2017**author** Benoit GIELLY <benoit.gielly@gmail.com>**find\_phantom\_vertices** (*fix=False*)

Find meshes with incorrect amount of vertices.

**Parameters** **fix** (*bool*) – Apply a fix to remove phantom vertices when True.

**mesh\_combine\_and\_keep** (*nodes=None, visible=True*)  
Combine meshes and keep the original.

**mesh\_reorder\_callback** ()  
Reorder or remap vertices of selected mesh(es).

### 4.3.16 bgdev.utils.name

Utility methods that parses names.

**created** 27/02/2017

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**find\_conflicts** (*query=False*)  
Create two selection sets to store name conflicts.

**Parameters** **query** (*bool*) – Only query the nodes without creating sets.

**Returns** Two separate lists for conflicting transforms and shapes.

**Return type** tuple

**generate\_unique\_name** (*name*)  
Generate a unique name based on the given one.

**remove\_end\_digits** (*node, rename=True*)  
Remove digits at the end of a name.

**Parameters**

- **node** (*str*) – Name of the node to “cleanup”
- **rename** (*bool*) – If True, rename the node, else just return the cleaned up name

**Returns** Name of the node stripped out of end digits.

**Return type** str

**remove\_end\_digits\_callback** ()  
Call back *remove\_end\_digits()*.

**rename\_deformers** (*node=None*)  
Rename all the deformers in the history of the given node.

The name string will be build according to the deformerType and the node name

**Parameters** **node** (*str*) – The node you want to rename deformers attached to it

#### Example

```
>>> node = "L_arm_geo"
>>> deformerType = "skinCluster"
>>> name = L_arm_geo_skinCluster"
```

**rename\_deformers\_callback** ()  
Call back *rename\_deformers()*.

### 4.3.17 bgdev.utils.pivot

This module deals with pivot import/export.

**created** 04/01/2018

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**export\_to\_file** (*path*, *rules=None*, *base\_grp='modelHierarchy'*)  
Export pivots matrices.

**Parameters**

- **path** (*file*) – Path to JSON file
- **rules** (*dict*) – Custom “SkinMap” or it will be autogenerated by `generate_data()`
- **base\_grp** (*str*) – Top group to pass into the data autogeneration function

**Returns** The pivot data

**Return type** dict

**generate\_data** (*base\_grp='modelHierarchy'*)  
Autogenerate a dictionary based on constraint nodes in scenes.

The key/value pair is equal to nodes/joints.

**Parameters** **base\_grp** (*str*) – Top group to investigate all descendents from.

**Returns** Dictionary that maps the node with its driver.

**Return type** dict

**import\_from\_file** (*path*)  
Import pivots from json file.

**Parameters** **path** (*file*) – Path to JSON file

**Returns** The data dictionary used

**Return type** dict

**reset\_pivot** (*mesh*, *matrix*)  
Reset given mesh to given matrix.

**Parameters**

- **mesh** (*str*) – Node to reset pivot
- **matrix** (*matrix*) – Matrix table giving the exact position of the pivot

### 4.3.18 bgdev.utils.rivet

Utility methods for rivets.

**created** 15/01/2019

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**follicle\_rivet** (*vertex=None*, *name=None*)  
Create a rivetted locator using follicle.

**Parameters**

- **vertex** (*str*) – Specify a vertex to attach the rivet to. Use selection otherwise.

- **name** (*str*) – Using this name for the rivet if passed, otherwise create one using the mesh name.

**Returns** The rivet locator created.

**Return type** *str*

**follicle\_rivet\_callback** ()

Create rivets based on selection.

### 4.3.19 bgdev.utils.scripts

Random scripts and tools (needs to be sorted...).

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**align\_fingers** (*nodes=None, aim\_vector=(1, 0, 0), up\_vector=(0, 1, 0), gui=False*)

Align selected locators based on first and last plane (use for fingers).

**find\_bad\_meshes** ()

Find meshes with incorrect amount of vertices.

**get\_vectors\_dialog** ()

Query aim and up vectors using a LayoutPromptDialog.

**logging\_time** ()

Print the logging time.

**mirror\_position** (*selected=False*)

Mirror selected along world Z+.

**quick\_distance** ()

Get distance between two nodes.

**set\_default\_clip\_plane** ()

Create a quick GUI to change de camera clipPlane values option vars.

**set\_near\_clip\_plane** ()

Set nearClipPlane to 1.

**show\_cam\_clip\_planes** ()

Turn nearClip and farClip planes visible in the CB for all cameras.

**show\_joint\_orient** (*value=True*)

Display jointOrient attributes in channel box.

**simple\_snap** (*srt=None*)

Snap quickly.

**snap** (*\*\*kwargs*)

Snap an obj on a target.

**time\_me** (*func*)

Use this decorator to time functions execution.

**toggle\_side\_select** (*toggle=True*)

Toggle selection.

### 4.3.20 bgdev.utils.serialize

Utility methods for data serialization (JSON, YAML, PKL, etc...).

**created** 26/04/2018

**author** Benoit Gielly <[benoit.gielly@gmail.com](mailto:benoit.gielly@gmail.com)>

**json\_dump** (*data*, *path*, *\*\*kwargs*)  
Export JSON file.

**Parameters**

- **data** (*dict*) – Dictionary to export as JSON
- **path** (*str*) – JSON file path to save data
- **kwargs** (*dict*) – Any extra flags to pass in the `json.dump()` command

**Returns** given file path

**Return type** `str`

**json\_load** (*path*, *\*\*kwargs*)  
Import JSON file.

**Parameters**

- **path** (*str*) – JSON file path to save data
- **kwargs** (*dict*) – Any extra flags to pass in the `json.load()` command

**Returns** The loaded data.

**Return type** `dict`

**yaml\_dump** (*data*, *path*, *ordered=False*, *\*\*kwargs*)  
Export YAML file.

**Parameters**

- **data** (*dict*) – Dictionary to export as JSON
- **path** (*str*) – YAML file path to save data
- **ordered** (*bool*) – Dump data ordered when True.
- **kwargs** (*dict*) – Any extra flags to pass in the `json.dump()` command

**Returns** given file path

**Return type** `str`

**yaml\_load** (*path*, *ordered=False*, *\*\*kwargs*)  
Import YAML file.

**Parameters**

- **path** (*str*) – YAML file path to save data
- **ordered** (*bool*) – Load data in an `orderedDict` when True.
- **kwargs** (*dict*) – Any extra flags to pass in the `json.load()` command

**Returns** the loaded data

**Return type** `dict`



### 4.3.21 bgdev.utils.shape

Utility methods for shapes.

**created** 05/03/2017

**author** Benoit Gielly <benoit.gielly@gmail.com>

**get\_shapes** (*node*, *intermediate=False*, *exclusive=False*)

Get the shapes of given node.

#### Parameters

- **node** (*str*) – Node to query its shapes
- **intermediate** (*bool*) – Get intermediate shapes when True.
- **exclusive** (*bool*) – Only return the intermediate shapes if True. Please note that the intermediate flag must be True as well.

**Returns** The shapes found below given node.

**Return type** list

**update\_intermediate** (*source*, *target*)

Update shapeOrig of target with source shape.

This tool also works on nubrsCurve and any deformableShapes. It will only replace one with another.

#### Parameters

- **source** (*str*) – source node
- **target** (*str*) – target node

**update\_intermediate\_callback** ()

Call back update\_intermediate().

**update\_plug** (*source\_shape*, *target\_shape*)

Update the plug.

#### Parameters

- **source\_shape** (*str*) – Name of the source shape.
- **target\_shape** (*str*) – Name of the target shape to update.

### 4.3.22 bgdev.utils.side

Module to deal with sided data.

**created** 02/07/2018

**author** Benoit Gielly <benoit.gielly@gmail.com>

**convert\_side** (*data*, *sides='LR'*, *both=False*)

Replace recursively all iterations of { } with side in a data structure.

#### Parameters

- **data** (*list or dict*) – The data to traverse and update.
- **sides** (*str*) – The sides to replace brackets with.
- **both** (*bool*) – By default, if the key is “sided”, the values associated will only get the same side as the key. Otherwise, values will get both sides. Default is *False*.

### Example

```
data = side.convert_side({
    "spine_01_grp": "spine_01_jnt",
    "{}_shoulder_mechanic_grp": "{}_arm_shoulders_jnt",
    "{}_knee_pistons_grp": "{}_leg_knee_jnt",
})

>>> # Result: {"L_knee_pistons_grp": "L_leg_knee_jnt",
    "L_shoulder_mechanic_grp": "L_arm_shoulders_jnt",
    "R_knee_pistons_grp": "R_leg_knee_jnt",
    "R_shoulder_mechanic_grp": "R_arm_shoulders_jnt",
    "spine_01_grp": "spine_01_jnt"} #
```

**Returns** An updated and sided copy of the given data.

**Return type** dict or list

## 4.3.23 bgdev.utils.skincluster

Utility methods for skinclusters.

**created** 26/02/2017

**author** Benoit Gielly <benoit.gielly@gmail.com>

**add\_influences** (*node*, *joints*)

Add given joints to the skincluster.

#### Parameters

- **node** (*str*) – Can be either the skincluster or the bound node.
- **joints** (*list*) – List of joints/influences to add.

**Raises** **RuntimeError** – If the skincluster cannot be found.

**add\_influences\_callback** ()

Call back *add\_influences* ().

**bind\_skincluster** (*mesh=None*, *bones=None*)

Create a new skincluster with preset options.

#### Parameters

- **mesh** (*str*) – Name of the geometry to bind.
- **bones** (*list*) – List of bones/influences.

**Returns** The created skincluster.

**Return type** str

**bind\_skincluster\_callback** ()

Call back *bind\_skincluster* ().

**copy\_skincluster** (*source*, *target*, *method='closestPoint'*)

Copy source skincluster onto target and keep same weights.

#### Parameters

- **source** (*str*) – source mesh to copy skincluster from

- **target** (*str*) – target mesh to paste skincluster to
- **method** (*str*) – influence association method. Default is `closestPoint`.

**Raises `RuntimeError`** – If the source skincluster cannot be found.

**copy\_skincluster\_callback** (*method*='closestPoint')

Call back `copy_skincluster()`.

**detach\_skincluster** (*node*, *clean*=False)

Detach the skin cluster attached to given node.

#### Parameters

- **node** (*str*) – Name of the node to unbind.
- **clean** (*bool*) – If True, remove shapeOrigs and unlock transforms.

**detach\_skincluster\_callback** (*clean*=False)

Call back `detach_skincluster()`.

**get\_influences** (*node*, *weighted*=False)

Get influences of given skincluster.

#### Parameters

- **node** (*str*) – Can be either the skincluster or the bound node.
- **weighted** (*bool*) – If True, returns only weighted influences.

**Returns** List of influences.

**Return type** list

**Raises `RuntimeError`** – If the skincluster cannot be found.

**get\_skincluster** (*node*)

Return the skincluster of given node.

If the passed node already is a skincluster, just return it.

**Parameters** **node** (*str*) – Can be either the skincluster or the bound node.

**Raises `RuntimeError`** – If the skincluster cannot be found.

**Returns** The skincluster bound to the node.

**Return type** str

**remove\_influences** (*node*, *joints*=None, *unused*=False, *disconnect*=False)

Remove given joints from the skincluster.

#### Parameters

- **node** (*str*) – Can be either the skincluster or the bound node.
- **joints** (*list*) – List of joints/influences to add.
- **unused** (*bool*) – Removed all unused influences found. This flag ignores *joints* when used.

**Raises `RuntimeError`** – If the skincluster cannot be found.

**remove\_influences\_callback** (*unused*=False)

Call back `remove_influences()`.

**reset\_skincluster** (*node*)

Reset the skin cluster in place.

**Parameters** **node** (*str*) – SkinCluster node or the geometry bound to it.

**reset\_skincluster\_callback** ()  
Call back *reset\_skincluster* ().

**select\_influences\_callback** ()  
Call back *select\_influences* ().

### 4.3.24 bgdev.utils.sort

Utility methods used to sort nodes and graphs.

**created** 05/03/2017

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**sort\_children** (*parent=None*)  
Sort children of the given parent alphabetically.

**Parameters** **parent** (*str*) – Name of the parent to use as a starting point.

**sort\_descendants** ()  
Sort all descendant children alphabetically.

**sort\_selection** ()  
Sort all children node alphabetically.

### 4.3.25 bgdev.utils.transform

Utility methods for transforms.

**created** 16/02/2018

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**reset\_group** (*node, suffix='\_grp'*)  
Create a transform node to reset the value of the selected object.

**Parameters**

- **node** (*str*) – name of node to reset
- **suffix** (*str*) – suffix to use for the reset group name

**Returns** reset group

**Return type** str

**reset\_groups** (*suffix='\_grp'*)  
Run *reset\_group* () on a selection of multiple nodes.

**Parameters** **suffix** (*str*) – suffix to use for the reset group name

**Returns** all reset group created

**Return type** list

### 4.3.26 bgdev.utils.vector

Utility methods to help dealing with vectors.

**created** 10/10/2016

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**aim\_in\_plane** (*positions*, *aim\_vector*=(1, 0, 0), *up\_vector*=(0, 1, 0))

Align selected locators based on plane made of the first and last.

**from\_euler** (*rotation*, *translate*=(0, 0, 0), *radians*=False)

Convert euler rotation into 3-axis matrix.

#### Parameters

- **rotation** (*tuple*) – Rotation values to add to the matrix table.
- **translate** (*tuple*) – Translation values to add to the matrix table.
- **radians** (*bool*) – If True, converts degrees to radians.

**Returns** Matrix of given euler rotates, with translate if given.

**Return type** list

**get\_closest\_point** (*source*, *targets*, *furthest*=False)

Find the closest node to the source of each targets.

#### Parameters

- **source** (*str*) – source node to use as starting point for distance calculation.
- **targets** (*list*) – each nodes to process.
- **furthest** (*bool*) – If True, gets the furthest node instead.

**Returns** the target node that's the closest to the source.

**Return type** str

**get\_distance\_between** (*node1*, *node2*, *distance\_between*=False, *bounding\_box*=False, *rotate\_pivot*=False)

Get the distance between two objects.

#### Parameters

- **node1** (*str*) – Node that determines start position
- **node2** (*str*) – Node that determines end position
- **distance\_between** (*bool*) – If True, creates a distance\_between node, query its value and delete it.
- **bounding\_box** (*bool*) – If True, creates a distance\_between node,
- **rotate\_pivot** (*bool*) – If True, creates a distance\_between node,

**Returns** distance between two given nodes.

**Return type** float

**get\_manipulator\_xforms** (*as\_matrix*=False)

Query the manipulator position and orientation.

**Parameters** **as\_matrix** (*bool*) – if True, returns a as\_matrix built from manip xforms.

**Returns**

**list of “XYZ” position and rotation values** or matrix array if *as\_matrix* is True.

**Return type** list

**get\_matrix\_from\_nodes** (*nodes*, *middle*=True, *aim\_vector*=(1, 0, 0), *up\_vector*=(0, 1, 0))

Return a matrix based on given nodes.

If passed nodes are 1 or more than 3, it simply return the manipulator position as a matrix. Otherwise, it'll use the second node as the aim axis and the third as up.

**Parameters**

- **nodes** (*list*) – list of nodes to get matrix
- **middle** (*bool*) – snap in between nodes 1 and 2 if True, else on first.
- **aim\_vector** (*tuple*) – default aim vector for the aimConstraint.
- **up\_vector** (*tuple*) – default up vector for the aimConstraint.

**Returns** matrix array.

**Return type** list

**get\_matrix\_from\_transforms** (*position*, *normal*, *tangent*)

Construct an MMatrix from position, normal and tangent.

**Parameters**

- **position** (*list*) – XYZ position.
- **normal** (*list*) – The normal vector used to compute rotation.
- **tangent** (*list*) – The tangent vector used to compute rotation.

**Returns** The MMatrix array.

**Return type** MMatrix

**get\_vectors** (*nodes*, *mode*='xform')

Generate world position vectors of each given nodes.

**Parameters**

- **nodes** (*list*) – list of nodes to return position as vector.
- **mode** (*str*) – choose between default “xform” or “pivot” to get world position.

**Yields** *maya.api.OpenMaya.MVector* – MVector of the node's world position

### 4.3.27 bgdev.utils.weightmap

Utility method to deal with weightmaps.

**created** 13/12/2018

**author** Benoit GIELLY <benoit.gielly@gmail.com>

**check\_libraries** ()

Check if ngSkinTools package is available.

**Returns** A mapping dict with the required ngSkinTool modules.

**Return type** dict

**Raises**

- **ImportError** – If the package can't be imported.

- **RuntimeError** – If the package can't be imported.

**export\_layer\_data** (*node, data, path*)  
Export ngLayerData into JSON file.

#### Parameters

- **node** (*str*) – Name of the mesh. Used as filename.
- **data** (*dict*) – The ngLayer dictionary to export.
- **path** (*str*) – The parent folder where the file will be saved into.

**export\_multiple\_skinweights** (*nodes, path='weights'*)  
Export multiple skinweights into a single file.

#### Parameters

- **nodes** (*str*) – List of nodes to export their skinweights.
- **path** (*str*) – File path to export weights.

**export\_skinweights** (*node, path='weights', export=True*)  
Export the weights of the given skincluster node.

#### Parameters

- **node** (*str*) – Name of the skinned node to export its weights.
- **path** (*str*) – File path to export the weights.
- **export** (*bool*) – Export weights into a file. Defaults to True. Use False when you only want the weights data.

**Returns** The weights data.

**Return type** dict

**export\_skinweights\_callback** ()  
Call back [\*export\\_skinweights\(\)\*](#).

**import\_layer\_data** (*node, path*)  
Import ngLayerData from JSON file.

#### Parameters

- **node** (*str*) – Name of the mesh. Used to find the JSON file.
- **path** (*str*) – The parent folder where the file is saved.

**Returns** The raw ngLayer data (somehow, this is a string!)

**Return type** str

**import\_multiple\_skinweights** (*path, layers=True*)  
Import multiple skinweights into a single file.

#### Parameters

- **path** (*str*) – File path to export weights.
- **layers** (*bool*) – Keep ngSkinTools layers or not.

**import\_skinweights** (*node, path='weights', data=None, layers=True, mode='vertexID'*)  
Export the weights of the given skincluster node.

#### Parameters

- **node** (*str*) – Name of the skinned node to export its weights.

- **path** (*str*) – File path to export the weights.
- **data** (*dict*) – Use this data directly instead of reading if from file.
- **layers** (*bool*) – Keep ngSkinTools layers or not.
- **mode** (*str*) – Can be “vertexID”, “UV” or “closestPoint”. Defaults to “vertexID”.

**import\_skinweights\_callback** ()  
Call back *import\_skinweights* ().

**initialize\_layers** (*node*)  
Remove ngSkinTools layers.

**Parameters** **node** (*str*) – Name of the mesh with a skinCluster attached.

**Returns** True if layers were created, false otherwise.

**Return type** bool

**remove\_layers** (*node*)  
Remove ngSkinTools layers.

**Parameters** **node** (*str*) – Name of the mesh with a skinCluster attached.

## 4.4 bgdev.cfg

Config module to pass data accross the repository project.

**Warning:** NEVER RELOAD THIS MODULE. If you do you, you will lose all the data that was stored by the application instance.

**created** 12/12/2018

**author** Benoit GIELLY <[benoit.gielly@gmail.com](mailto:benoit.gielly@gmail.com)>

## 4.5 bgdev.docstring

Docstring template using Google convention.

**created** 16/11/2018

**author** Benoit Gielly <[benoit.gielly@gmail.com](mailto:benoit.gielly@gmail.com)>

This module is just a template to show how to write proper docstrings using the Google’s syntax. It will show you all the different applications and keywords you can use so sphinx can generate a nice looking documentation!

**You can find a link to the Google Docstring guide here:** [http://sphinxcontrib-napoleon.readthedocs.io/en/latest/example\\_google.html](http://sphinxcontrib-napoleon.readthedocs.io/en/latest/example_google.html)

Happy docstring-ing! :D

**class ExampleClass** (*param1, param2, param3*)  
Bases: object

The summary line for a class docstring should fit on one line.



If the class has public attributes, they may be documented here in an `Attributes` section and follow the same formatting as a function's `Args` section. Alternatively, attributes may be documented inline with the attribute's declaration (see `__init__` method below).

Properties created with the `@property` decorator should be documented in the property's getter method.

**attr1**

Description of *attr1*.

**Type** str

**attr2**

Description of *attr2*.

**Type** int, optional

**Parameters**

- **param1** (*str*) – Description of *param1*.
- **param2** (*int*) – Description of *param2*. Multiple lines are supported.
- **param3** (*list(str)*) – Description of *param3*.

---

**Note:** Do not include the *self* parameter in the `Args` section.

---

**\_\_init\_\_** (*param1*, *param2*, *param3*)

Class constructor.

You should avoid documenting classes in the `__init__` method of the class. It should go within the class docstring instead.

**attr3**

Doc comment *inline* with attribute

**attr4**

Doc comment *before* attribute, with type specified

**Type** list(str)

**attr5**

Docstring *after* attribute, with type specified.

**Type** str

**example\_method** (*param1*, *param2*)

Class methods are similar to regular functions.

See [example\\_function\(\)](#) for a better description.

---

**Note:** Do not include the *self* parameter in the `Args` section.

---

**property getter\_only\_property**

Properties should be documented in their getter method.

**Type** str

**property getter\_setter\_property**

Get a property.

If the setter method contains notable behavior, it should be mentioned here.

**Parameters**

- **value** (*str*) – Properties with both a getter and setter
- **only be documented in their getter method.** (*should*) –

**Returns** The returned value/object.

**Return type** list

**example\_decorated\_function** (*arg*)

Test the docstring to ensure the @wraps decorator worked.

**Parameters** **arg** (*str*) – Description of *arg1*

**Returns** The first argument *arg1*

**Return type** type

**example\_decorator** (*func*)

Decorate a method.

Decorators need a special treatment to generate the documentation properly. You have to decorate the sub-function with the *wraps* method of the *functools* module. If you don't, the docstring of the decorated function will be skipped.

**Example**

```
from functools import wraps

def example_decorator(func):
    "Decorator docstring"

    @wraps(func)
    def function(*args, **kwargs):
        returned_func = func(*args, **kwargs)
        return returned_func
    return function
```

**example\_function** (*param1*, *param2=None*, \**args*, \*\**kwargs*)

Show an example of a module level function.

Function parameters should be documented in the Args section. The name of each parameter is required. The type and description of each parameter is optional, but should be included if not obvious.

If \**args* or \*\**kwargs* are accepted, they should be listed as \**args* and \*\**kwargs*.

The format for a parameter is:

```
name (type): description
    The description may span multiple lines. Following
    lines should be indented. The "(type)" is optional.

    Multiple paragraphs are supported in parameter
    descriptions.
```

**Parameters**

- **param1** (*int*) – The first parameter.
- **param2** (*str*) – The second parameter. Defaults to None. Second line of description should be indented.

- **\*args** – Variable length argument list.
- **\*\*kwargs** – Arbitrary keyword arguments.

### Returns

True if successful, False otherwise.

The return type is optional and may be specified at the beginning of the Returns section followed by a colon.

The Returns section may span multiple lines and paragraphs. Following lines should be indented to match the first line.

The Returns section supports any reStructuredText formatting, including literal blocks:

```
{
    'param1': param1,
    'param2': param2
}
```

### Return type bool

### Raises

- **AttributeError** – The Raises section is a list of all exceptions that are relevant to the interface.
- **ValueError** – If *param2* is equal to *param1*.

### Example

Examples should be written in doctest format, and should illustrate how to use the function.

```
>>> print([i for i in range(4)])
[0, 1, 2, 3]
```

### Examples

You can also use literal blocks:

```
print([i for i in range(4)])
>>> [0, 1, 2, 3]
```

### Todo:

- The Todo section lists in an orange block every task that needs to be done.
- Make sure to use an \* (asterisk) to display bullet points

### example\_function\_with\_types(*arg1*, *arg2*)

Show an example function with types documented in the docstring.

PEP 484 type annotations are supported. If attribute, parameter, and return types are annotated according to PEP 484, they do not need to be included in the docstring:

### Parameters

- **arg1** (*int*) – The first parameter.

- **arg2** (*str*) – The second parameter.

**Returns** The return value. True for success, False otherwise.

**Return type** bool

**example\_generator** (*num*)

Create generators.

They have a `Yields` section instead of a `Returns` section.

**Parameters** **num** (*int*) – The upper limit of the range to generate, from 0 to *num* - 1.

**Yields** *int* – The next number in the range of 0 to *num* - 1.

## RELEASE NOTES

### 5.1 1.0.0 (2020-12-10)

- First public release of the repo



**MODULE INDEX**





---

**CHAPTER  
SEVEN**

---

**INDEX**



## PYTHON MODULE INDEX

### b

- bgdev, 9
- bgdev.cfg, 44
- bgdev.docstring, 44
- bgdev.tools, 9
  - bgdev.tools.atom\_import\_export, 14
  - bgdev.tools.bindpose, 14
  - bgdev.tools.curve, 15
  - bgdev.tools.mayaColorPicker, 9
  - bgdev.tools.mayaColorPicker.core, 9
  - bgdev.tools.mayaColorPicker.main, 9
  - bgdev.tools.mel2py, 15
  - bgdev.tools.outliner\_color, 16
  - bgdev.tools.performance, 16
  - bgdev.tools.pluginManager, 9
  - bgdev.tools.pluginManager.core, 9
  - bgdev.tools.pypredef, 17
  - bgdev.tools.renamer, 10
    - bgdev.tools.renamer.core, 10
    - bgdev.tools.renamer.ui, 10
  - bgdev.tools.renamer.utils, 10
  - bgdev.tools.symmetry, 18
  - bgdev.tools.symmetry\_api, 19
  - bgdev.tools.toolbar, 11
    - bgdev.tools.toolbar.cfg, 12
    - bgdev.tools.toolbar.main, 12
    - bgdev.tools.toolbar.tabs, 11
      - bgdev.tools.toolbar.tabs.maya, 11
      - bgdev.tools.toolbar.tabs.maya.main, 11
      - bgdev.tools.toolbar.tabs.system, 11
      - bgdev.tools.toolbar.tabs.system.main, 11
    - bgdev.tools.toolbar.ui, 12
  - bgdev.tools.toolbar.utils, 12
  - bgdev.tools.uniloop, 20
  - bgdev.tools.vector\_plane, 21
- bgdev.ui, 21
  - bgdev.ui.dock, 21
  - bgdev.ui.flow\_layout, 21
  - bgdev.ui.progress\_bar, 21
  - bgdev.ui.utils, 22
  - bgdev.ui.widgets, 23
  - bgdev.utils, 23
    - bgdev.utils.attribute, 23
    - bgdev.utils.bindpose, 24
    - bgdev.utils.color, 24
    - bgdev.utils.connections, 25
    - bgdev.utils.constraints, 25
    - bgdev.utils.controller, 25
    - bgdev.utils.curve, 26
    - bgdev.utils.decorator, 27
    - bgdev.utils.deformer, 29
    - bgdev.utils.display, 30
    - bgdev.utils.history, 30
    - bgdev.utils.hotkey, 31
    - bgdev.utils.joints, 31
    - bgdev.utils.locators, 32
    - bgdev.utils.mesh, 32
    - bgdev.utils.name, 33
    - bgdev.utils.pivot, 34
    - bgdev.utils.rivet, 34
    - bgdev.utils.scripts, 35
    - bgdev.utils.serialize, 36
    - bgdev.utils.shape, 37
    - bgdev.utils.side, 37
    - bgdev.utils.skincluster, 38
    - bgdev.utils.sort, 40
    - bgdev.utils.transform, 40
    - bgdev.utils.vector, 41
    - bgdev.utils.weightmap, 42



## Symbols

`__init__()` (*ColorPicker* method), 9  
`__init__()` (*ExampleClass* method), 45  
`__init__()` (*Profiler* method), 16  
`__init__()` (*ProgressBar* method), 21  
`__init__()` (*ProgressBar2* method), 22  
`__init__()` (*Symmetry* method), 18, 19

## A

`add_influences()` (in *module* *bgdev.utils.skincluster*), 38  
`add_influences_callback()` (in *module* *bgdev.utils.skincluster*), 38  
`add_joints()` (in *module* *bgdev.utils.joints*), 31  
`aim_in_plane()` (in *module* *bgdev.utils.vector*), 41  
`aim_to_children()` (in *module* *bgdev.utils.constraints*), 25  
`align_fingers()` (in *module* *bgdev.utils.scripts*), 35  
`atom_export()` (in *module* *bgdev.tools.atom\_import\_export*), 14  
`atom_import()` (in *module* *bgdev.tools.atom\_import\_export*), 14  
`attach_locators_to_curve()` (in *module* *bgdev.utils.locators*), 32  
`attach_nodes_to_curve()` (in *module* *bgdev.tools.curve*), 15  
`attr1` (*ExampleClass* attribute), 45  
`attr2` (*ExampleClass* attribute), 45  
`attr3` (*ExampleClass* attribute), 45  
`attr4` (*ExampleClass* attribute), 45  
`attr5` (*ExampleClass* attribute), 45  
`attributes_restore()` (in *module* *bgdev.utils.attribute*), 23  
`attributes_toggle()` (in *module* *bgdev.utils.attribute*), 24  
`attributes_unlock()` (in *module* *bgdev.utils.attribute*), 24

## B

`bgdev`  
*module*, 9  
`bgdev.cfg`

*module*, 44  
`bgdev.docstring`  
*module*, 44  
`bgdev.tools`  
*module*, 9  
`bgdev.tools.atom_import_export`  
*module*, 14  
`bgdev.tools.bindpose`  
*module*, 14  
`bgdev.tools.curve`  
*module*, 15  
`bgdev.tools.mayaColorPicker`  
*module*, 9  
`bgdev.tools.mayaColorPicker.core`  
*module*, 9  
`bgdev.tools.mayaColorPicker.main`  
*module*, 9  
`bgdev.tools.mel2py`  
*module*, 15  
`bgdev.tools.outliner_color`  
*module*, 16  
`bgdev.tools.performance`  
*module*, 16  
`bgdev.tools.pluginManager`  
*module*, 9  
`bgdev.tools.pluginManager.core`  
*module*, 9  
`bgdev.tools.pypredef`  
*module*, 17  
`bgdev.tools.renamer`  
*module*, 10  
`bgdev.tools.renamer.core`  
*module*, 10  
`bgdev.tools.renamer.ui`  
*module*, 10  
`bgdev.tools.renamer.utils`  
*module*, 10  
`bgdev.tools.symmetry`  
*module*, 18  
`bgdev.tools.symmetry_api`  
*module*, 19  
`bgdev.tools.toolbar`

- module, 11
- bgdev.tools.toolbar.cfg
  - module, 12
- bgdev.tools.toolbar.main
  - module, 12
- bgdev.tools.toolbar.tabs
  - module, 11
- bgdev.tools.toolbar.tabs.maya
  - module, 11
- bgdev.tools.toolbar.tabs.maya.main
  - module, 11
- bgdev.tools.toolbar.tabs.system
  - module, 11
- bgdev.tools.toolbar.tabs.system.main
  - module, 11
- bgdev.tools.toolbar.ui
  - module, 12
- bgdev.tools.toolbar.utils
  - module, 12
- bgdev.tools.uniloop
  - module, 20
- bgdev.tools.vector\_plane
  - module, 21
- bgdev.ui
  - module, 21
- bgdev.ui.dock
  - module, 21
- bgdev.ui.flow\_layout
  - module, 21
- bgdev.ui.progress\_bar
  - module, 21
- bgdev.ui.utils
  - module, 22
- bgdev.ui.widgets
  - module, 23
- bgdev.utils
  - module, 23
- bgdev.utils.attribute
  - module, 23
- bgdev.utils.bindpose
  - module, 24
- bgdev.utils.color
  - module, 24
- bgdev.utils.connections
  - module, 25
- bgdev.utils.constraints
  - module, 25
- bgdev.utils.controller
  - module, 25
- bgdev.utils.curve
  - module, 26
- bgdev.utils.decorator
  - module, 27
- bgdev.utils.deformer

- module, 29
- bgdev.utils.display
  - module, 30
- bgdev.utils.history
  - module, 30
- bgdev.utils.hotkey
  - module, 31
- bgdev.utils.joints
  - module, 31
- bgdev.utils.locators
  - module, 32
- bgdev.utils.mesh
  - module, 32
- bgdev.utils.name
  - module, 33
- bgdev.utils.pivot
  - module, 34
- bgdev.utils.rivet
  - module, 34
- bgdev.utils.scripts
  - module, 35
- bgdev.utils.serialize
  - module, 36
- bgdev.utils.shape
  - module, 37
- bgdev.utils.side
  - module, 37
- bgdev.utils.skincluster
  - module, 38
- bgdev.utils.sort
  - module, 40
- bgdev.utils.transform
  - module, 40
- bgdev.utils.vector
  - module, 41
- bgdev.utils.weightmap
  - module, 42
- bind\_skincluster() (in module *bgdev.utils.skincluster*), 38
- bind\_skincluster\_callback() (in module *bgdev.utils.skincluster*), 38

## C

- center() (*Symmetry property*), 18, 19
- check\_image() (in module *bgdev.tools.renamer.utils*), 10
- check\_image() (in module *bgdev.tools.toolbar.utils*), 12
- check\_image() (in module *bgdev.ui.utils*), 22
- check\_libraries() (in module *bgdev.utils.weightmap*), 42
- check\_modpath\_has\_init() (in module *bgdev.tools.toolbar.utils*), 12

- cleanup\_flags() (in module *bgdev.tools.pypredef*), 17
- color() (*ColorPicker* property), 9
- ColorPicker (class in *bgdev.tools.mayaColorPicker.main*), 9
- convert\_command() (in module *bgdev.tools.mel2py*), 15
- convert\_css\_color() (in module *bgdev.utils.color*), 24
- convert\_file() (in module *bgdev.tools.mel2py*), 15
- convert\_side() (in module *bgdev.utils.side*), 37
- copy\_skincluster() (in module *bgdev.utils.skincluster*), 38
- copy\_skincluster\_callback() (in module *bgdev.utils.skincluster*), 39
- create\_aim\_locator() (in module *bgdev.utils.locators*), 32
- create\_bindpose() (in module *bgdev.tools.bindpose*), 14
- create\_bindpose() (in module *bgdev.utils.bindpose*), 24
- create\_control() (in module *bgdev.utils.controller*), 25
- create\_hotkey() (in module *bgdev.utils.hotkey*), 31
- create\_hotkeys\_from\_yaml() (in module *bgdev.utils.hotkey*), 31
- create\_module\_callback() (in module *bgdev.tools.toolbar.utils*), 12
- create\_separator\_attribute() (in module *bgdev.utils.attribute*), 24
- curve\_from\_nodes() (in module *bgdev.tools.curve*), 15
- curve\_interpolate() (in module *bgdev.utils.curve*), 26
- D**
- delete\_history() (in module *bgdev.utils.history*), 30
- delete\_unused() (in module *bgdev.utils.history*), 30
- detach\_skincluster() (in module *bgdev.utils.skincluster*), 39
- detach\_skincluster\_callback() (in module *bgdev.utils.skincluster*), 39
- dock\_widget() (in module *bgdev.tools.toolbar.tabs.maya.main*), 11
- dock\_widget() (in module *bgdev.ui.dock*), 21
- E**
- end() (*ProgressBar* method), 22
- end() (*ProgressBar2* method), 22
- example\_decorated\_function() (in module *bgdev.docstring*), 46
- example\_decorator() (in module *bgdev.docstring*), 46
- example\_function() (in module *bgdev.docstring*), 46
- example\_function\_with\_types() (in module *bgdev.docstring*), 47
- example\_generator() (in module *bgdev.docstring*), 48
- example\_method() (*ExampleClass* method), 45
- ExampleClass (class in *bgdev.docstring*), 44
- export\_dialog() (in module *bgdev.tools.atom\_import\_export*), 14
- export\_layer\_data() (in module *bgdev.utils.weightmap*), 43
- export\_multiple\_skinweights() (in module *bgdev.utils.weightmap*), 43
- export\_skinweights() (in module *bgdev.utils.weightmap*), 43
- export\_skinweights\_callback() (in module *bgdev.utils.weightmap*), 43
- export\_to\_file() (in module *bgdev.utils.pivot*), 34
- F**
- find\_bad\_meshes() (in module *bgdev.utils.scripts*), 35
- find\_conflicts() (in module *bgdev.utils.name*), 33
- find\_phantom\_vertices() (in module *bgdev.utils.mesh*), 32
- follicle\_rivet() (in module *bgdev.utils.rivet*), 34
- follicle\_rivet\_callback() (in module *bgdev.utils.rivet*), 35
- frames\_per\_second() (in module *bgdev.tools.performance*), 16
- freeze\_transforms() (in module *bgdev.utils.history*), 31
- from\_euler() (in module *bgdev.utils.vector*), 41
- G**
- generate\_data() (in module *bgdev.utils.pivot*), 34
- generate\_pypredef() (in module *bgdev.tools.pypredef*), 17
- generate\_qrc() (in module *bgdev.ui.utils*), 22
- generate\_unique\_name() (in module *bgdev.utils.name*), 33
- get\_all\_controls() (in module *bgdev.tools.bindpose*), 14
- get\_all\_controls() (in module *bgdev.utils.controller*), 26
- get\_available\_controls() (in module *bgdev.ui.dock*), 21
- get\_class\_name() (in module *bgdev.tools.toolbar.main*), 12
- get\_closest\_point() (in module *bgdev.tools.curve*), 15
- get\_closest\_point() (in module *bgdev.utils.vector*), 41

[get\\_ctrl\\_data\(\)](#) (in module [bgdev.utils.controller](#)), 26  
[get\\_dcc\(\)](#) (in module [bgdev.tools.toolbar.utils](#)), 13  
[get\\_dcc\\_package\(\)](#) (in module [bgdev.tools.toolbar.utils](#)), 13  
[get\\_distance\\_between\(\)](#) (in module [bgdev.utils.vector](#)), 41  
[get\\_influences\(\)](#) (in module [bgdev.utils.skincluster](#)), 39  
[get\\_manipulator\\_xforms\(\)](#) (in module [bgdev.utils.vector](#)), 41  
[get\\_matrix\\_from\\_nodes\(\)](#) (in module [bgdev.utils.vector](#)), 42  
[get\\_matrix\\_from\\_transforms\(\)](#) (in module [bgdev.utils.vector](#)), 42  
[get\\_maya\\_urls\(\)](#) (in module [bgdev.tools.pyprdef](#)), 18  
[get\\_mirrorvector\(\)](#) (in module [bgdev.tools.vector\\_plane](#)), 21  
[get\\_rig\\_controls\(\)](#) (in module [bgdev.utils.controller](#)), 26  
[get\\_shapes\(\)](#) (in module [bgdev.utils.shape](#)), 37  
[get\\_side\\_vertices\(\)](#) ([Symmetry](#) method), 18, 19  
[get\\_sides\\_from\\_rich\(\)](#) ([Symmetry](#) static method), 19  
[get\\_skincluster\(\)](#) (in module [bgdev.utils.skincluster](#)), 39  
[get\\_vectors\(\)](#) (in module [bgdev.utils.vector](#)), 42  
[get\\_vectors\\_dialog\(\)](#) (in module [bgdev.utils.scripts](#)), 35  
[get\\_vertex\\_id\(\)](#) ([Symmetry](#) static method), 18, 20  
[get\\_vertex\\_name\(\)](#) ([Symmetry](#) method), 18, 20  
[get\\_workspaces\(\)](#) (in module [bgdev.tools.toolbar.tabs.maya.main](#)), 11  
[getter\\_only\\_property\(\)](#) ([ExampleClass](#) property), 45  
[getter\\_setter\\_property\(\)](#) ([ExampleClass](#) property), 45  
**I**  
[import\\_dialog\(\)](#) (in module [bgdev.tools.atom\\_import\\_export](#)), 14  
[import\\_from\\_file\(\)](#) (in module [bgdev.utils.pivot](#)), 34  
[import\\_layer\\_data\(\)](#) (in module [bgdev.utils.weightmap](#)), 43  
[import\\_multiple\\_skinweights\(\)](#) (in module [bgdev.utils.weightmap](#)), 43  
[import\\_skinweights\(\)](#) (in module [bgdev.utils.weightmap](#)), 43  
[import\\_skinweights\\_callback\(\)](#) (in module [bgdev.utils.weightmap](#)), 44  
[in\\_maya\(\)](#) (in module [bgdev.ui.utils](#)), 23  
[indent\(\)](#) (in module [bgdev.tools.pyprdef](#)), 18  
[initialize\\_layers\(\)](#) (in module [bgdev.utils.weightmap](#)), 44  
[is\\_cancelled\(\)](#) ([ProgressBar](#) method), 22  
[iskeyword\(\)](#) (in module [bgdev.tools.pyprdef](#)), 18  
**J**  
[json\\_dump\(\)](#) (in module [bgdev.utils.serialize](#)), 36  
[json\\_load\(\)](#) (in module [bgdev.utils.serialize](#)), 36  
**L**  
[left\(\)](#) ([Symmetry](#) property), 18, 20  
[list\\_controllers\(\)](#) (in module [bgdev.utils.controller](#)), 26  
[locator\\_on\\_selection\(\)](#) (in module [bgdev.utils.locators](#)), 32  
[logging\\_time\(\)](#) (in module [bgdev.utils.scripts](#)), 35  
**M**  
[main\\_window\(\)](#) (in module [bgdev.ui.utils](#)), 23  
[matrix\\_constraint\(\)](#) (in module [bgdev.utils.constraints](#)), 25  
[matrix\\_constraint\\_callback\(\)](#) (in module [bgdev.utils.constraints](#)), 25  
[maya\\_window\(\)](#) (in module [bgdev.tools.renamer.ui](#)), 10  
[maya\\_window\(\)](#) (in module [bgdev.tools.toolbar.tabs.maya.main](#)), 11  
[maya\\_window\(\)](#) (in module [bgdev.ui.utils](#)), 23  
[mesh\\_combine\\_and\\_keep\(\)](#) (in module [bgdev.utils.mesh](#)), 32  
[mesh\\_reorder\\_callback\(\)](#) (in module [bgdev.utils.mesh](#)), 33  
[method\\_decorator\(\)](#) (in module [bgdev.tools.renamer.utils](#)), 10  
[method\\_decorator\(\)](#) (in module [bgdev.utils.decorator](#)), 28  
[mirror\(\)](#) ([Symmetry](#) method), 19, 20  
[mirror\\_layout\(\)](#) (in module [bgdev.tools.vector\\_plane](#)), 21  
[mirror\\_position\(\)](#) (in module [bgdev.utils.scripts](#)), 35  
[mirror\\_selection\(\)](#) ([Symmetry](#) method), 19, 20  
[mirrorToolCMD\(\)](#) (in module [bgdev.tools.vector\\_plane](#)), 21  
[modpath\\_from\\_file\(\)](#) (in module [bgdev.tools.toolbar.utils](#)), 13  
**module**  
[bgdev](#), 9  
[bgdev.cfg](#), 44  
[bgdev.docstring](#), 44  
[bgdev.tools](#), 9  
[bgdev.tools.atom\\_import\\_export](#), 14  
[bgdev.tools.bindpose](#), 14  
[bgdev.tools.curve](#), 15



bgdev.tools.mayaColorPicker, 9  
 bgdev.tools.mayaColorPicker.core, 9  
 bgdev.tools.mayaColorPicker.main, 9  
 bgdev.tools.mel2py, 15  
 bgdev.tools.outliner\_color, 16  
 bgdev.tools.performance, 16  
 bgdev.tools.pluginManager, 9  
 bgdev.tools.pluginManager.core, 9  
 bgdev.tools.pypredef, 17  
 bgdev.tools.renamer, 10  
 bgdev.tools.renamer.core, 10  
 bgdev.tools.renamer.ui, 10  
 bgdev.tools.renamer.utils, 10  
 bgdev.tools.symmetry, 18  
 bgdev.tools.symmetry\_api, 19  
 bgdev.tools.toolbar, 11  
 bgdev.tools.toolbar.cfg, 12  
 bgdev.tools.toolbar.main, 12  
 bgdev.tools.toolbar.tabs, 11  
 bgdev.tools.toolbar.tabs.maya, 11  
 bgdev.tools.toolbar.tabs.maya.main, 11  
 bgdev.tools.toolbar.tabs.system, 11  
 bgdev.tools.toolbar.tabs.system.main, 11  
 bgdev.tools.toolbar.ui, 12  
 bgdev.tools.toolbar.utils, 12  
 bgdev.tools.uniloop, 20  
 bgdev.tools.vector\_plane, 21  
 bgdev.ui, 21  
 bgdev.ui.dock, 21  
 bgdev.ui.flow\_layout, 21  
 bgdev.ui.progress\_bar, 21  
 bgdev.ui.utils, 22  
 bgdev.ui.widgets, 23  
 bgdev.utils, 23  
 bgdev.utils.attribute, 23  
 bgdev.utils.bindpose, 24  
 bgdev.utils.color, 24  
 bgdev.utils.connections, 25  
 bgdev.utils.constraints, 25  
 bgdev.utils.controller, 25  
 bgdev.utils.curve, 26  
 bgdev.utils.decorator, 27  
 bgdev.utils.deformer, 29  
 bgdev.utils.display, 30  
 bgdev.utils.history, 30  
 bgdev.utils.hotkey, 31  
 bgdev.utils.joints, 31  
 bgdev.utils.locators, 32  
 bgdev.utils.mesh, 32  
 bgdev.utils.name, 33  
 bgdev.utils.pivot, 34  
 bgdev.utils.rivet, 34

bgdev.utils.scripts, 35  
 bgdev.utils.serialize, 36  
 bgdev.utils.shape, 37  
 bgdev.utils.side, 37  
 bgdev.utils.skincluster, 38  
 bgdev.utils.sort, 40  
 bgdev.utils.transform, 40  
 bgdev.utils.vector, 41  
 bgdev.utils.weightmap, 42

## N

node() (*ColorPicker* property), 9

## P

pairblend\_three\_nodes() (in module *bgdev.utils.connections*), 25  
 populate\_table() (*Symmetry* method), 19, 20  
 Profiler (class in *bgdev.tools.performance*), 16  
 ProgressBar (class in *bgdev.ui.progress\_bar*), 21  
 ProgressBar2 (class in *bgdev.ui.progress\_bar*), 22

## Q

quick\_connections() (in module *bgdev.utils.connections*), 25  
 quick\_distance() (in module *bgdev.utils.scripts*), 35

## R

real\_focus() (in module *bgdev.utils.display*), 30  
 real\_focus\_old() (in module *bgdev.utils.display*), 30  
 remove\_end\_digits() (in module *bgdev.utils.name*), 33  
 remove\_end\_digits\_callback() (in module *bgdev.utils.name*), 33  
 remove\_influences() (in module *bgdev.utils.skincluster*), 39  
 remove\_influences\_callback() (in module *bgdev.utils.skincluster*), 39  
 remove\_layers() (in module *bgdev.utils.weightmap*), 44  
 rename\_deformers() (in module *bgdev.utils.name*), 33  
 rename\_deformers\_callback() (in module *bgdev.utils.name*), 33  
 REPEAT() (in module *bgdev.utils.decorator*), 27  
 reset\_group() (in module *bgdev.utils.transform*), 40  
 reset\_groups() (in module *bgdev.utils.transform*), 40  
 reset\_pivot() (in module *bgdev.utils.pivot*), 34  
 reset\_skincluster() (in module *bgdev.utils.skincluster*), 39  
 reset\_skincluster\_callback() (in module *bgdev.utils.skincluster*), 40

[restore\\_bindpose\(\)](#) (in module [bgdev.tools.bindpose](#)), 14  
[restore\\_bindpose\(\)](#) (in module [bgdev.utils.bindpose](#)), 24  
[right\(\)](#) (*Symmetry property*), 19, 20

## S

[select\\_all\(\)](#) (in module [bgdev.utils.controler](#)), 26  
[select\\_influences\\_callback\(\)](#) (in module [bgdev.utils.skincluster](#)), 40  
[select\\_rig\\_controls\(\)](#) (in module [bgdev.utils.controler](#)), 26  
[selected\(\)](#) (in module [bgdev.utils.decorator](#)), 29  
[set\\_default\\_clip\\_plane\(\)](#) (in module [bgdev.utils.scripts](#)), 35  
[set\\_near\\_clip\\_plane\(\)](#) (in module [bgdev.utils.scripts](#)), 35  
[setMirrorToolCMD\(\)](#) (in module [bgdev.tools.vector\\_plane](#)), 21  
[show\(\)](#) (in module [bgdev.tools.renamer](#)), 10  
[show\(\)](#) (in module [bgdev.tools.renamer.core](#)), 10  
[show\(\)](#) (in module [bgdev.tools.toolbar.main](#)), 12  
[show\(\)](#) (in module [bgdev.tools.toolbar.tabs.maya.main](#)), 11  
[show\(\)](#) (in module [bgdev.tools.toolbar.tabs.system.main](#)), 11  
[show\\_cam\\_clip\\_planes\(\)](#) (in module [bgdev.utils.scripts](#)), 35  
[show\\_joint\\_orient\(\)](#) (in module [bgdev.utils.joints](#)), 32  
[show\\_joint\\_orient\(\)](#) (in module [bgdev.utils.scripts](#)), 35  
[simple\\_snap\(\)](#) (in module [bgdev.utils.scripts](#)), 35  
[snap\(\)](#) (in module [bgdev.utils.scripts](#)), 35  
[sort\\_children\(\)](#) (in module [bgdev.utils.sort](#)), 40  
[sort\\_descendants\(\)](#) (in module [bgdev.utils.sort](#)), 40  
[sort\\_selection\(\)](#) (in module [bgdev.utils.sort](#)), 40  
[start\(\)](#) (*Profiler method*), 16  
[start\(\)](#) (*ProgressBar method*), 22  
[start\(\)](#) (*ProgressBar2 method*), 22  
[stats\(\)](#) (*Profiler method*), 16  
[status\(\)](#) (*ProgressBar property*), 22  
[status\(\)](#) (*ProgressBar2 property*), 22  
[step\(\)](#) (*ProgressBar method*), 22  
[step\(\)](#) (*ProgressBar2 method*), 22  
[stop\(\)](#) (*Profiler method*), 16  
[Symmetry](#) (class in [bgdev.tools.symmetry](#)), 18  
[Symmetry](#) (class in [bgdev.tools.symmetry\\_api](#)), 19

## T

[time\\_me\(\)](#) (in module [bgdev.utils.scripts](#)), 35  
[to\\_qwidget\(\)](#) (in module [bgdev.tools.toolbar.tabs.maya.main](#)), 11

[to\\_qwidget\(\)](#) (in module [bgdev.ui.utils](#)), 23  
[toggle\\_side\\_select\(\)](#) (in module [bgdev.utils.scripts](#)), 35  
[transfer\\_base\\_weights\(\)](#) (in module [bgdev.utils.deformer](#)), 29  
[transfer\\_base\\_weights\\_api\(\)](#) (in module [bgdev.utils.deformer](#)), 29

## U

[UNDO\(\)](#) (in module [bgdev.tools.renamer.utils](#)), 10  
[UNDO\(\)](#) (in module [bgdev.utils.decorator](#)), 27  
[UNDO\\_REPEAT\(\)](#) (in module [bgdev.utils.decorator](#)), 28  
[update\(\)](#) (*Symmetry method*), 19, 20  
[update\\_intermediate\(\)](#) (in module [bgdev.utils.shape](#)), 37  
[update\\_intermediate\\_callback\(\)](#) (in module [bgdev.utils.shape](#)), 37  
[update\\_lattice\(\)](#) (in module [bgdev.utils.deformer](#)), 30  
[update\\_lattice\\_callback\(\)](#) (in module [bgdev.utils.deformer](#)), 30  
[update\\_plug\(\)](#) (in module [bgdev.utils.shape](#)), 37

## V

[valid\\_keyset\(\)](#) (in module [bgdev.utils.hotkey](#)), 31  
[vector\(\)](#) (in module [bgdev.tools.vector\\_plane](#)), 21  
[viewport\\_display\\_types\(\)](#) (in module [bgdev.utils.display](#)), 30

## Y

[yaml\\_dump\(\)](#) (in module [bgdev.tools.toolbar.utils](#)), 13  
[yaml\\_dump\(\)](#) (in module [bgdev.utils.serialize](#)), 36  
[yaml\\_load\(\)](#) (in module [bgdev.tools.toolbar.utils](#)), 13  
[yaml\\_load\(\)](#) (in module [bgdev.utils.serialize](#)), 36