
Pickrunner Documentation

Release 1.0

Colin Kennedy

Feb 10, 2018

1	Installation	3
1.1	Installing By Module	3
1.2	Installing Manually	3
2	Scene Setup	5
2.1	Assignment Mode	5
2.2	Selection Mode	6
3	Drawback To Pickrunner	7
4	Final Notes	9
4.1	API Documentation	9
	Python Module Index	13

Maya's pickWalk tool is okay, but it doesn't work well for many common scenarios. In particular, pickWalking is useless for rigs, because rig controllers are rarely grouped so that a user can move between them easily.

For this reason and a few others, Pickrunner was created.

Pickrunner is very simple. You assign how to move between objects in your scene once and then you can use your up/down/left/right keys to move between those objects just like you normally would. If an object doesn't have any Pickrunner data assigned to it, the tool will just use pickWalk, instead. That way, even if Pickrunner is only partially implemented in a scene, it won't interrupt the artist's process.

CHAPTER 1

Installation

1.1 Installing By Module

Download the pickrunner project from online.

```
git clone https://github.com/ColinKennedy/pickrunner.git
```

Pickrunner comes with a module file located in the “modules” folder called “pickrunner.mod”. Just add the full path to the “modules” folder to the `MAYA_MODULE_PATH` environment variable and restart Maya.

Pickrunner will build a brand new shelf for you to use its GUI.

The paths listed in `pickrunner.mod` are all relative to the main folder so, if you need to place the `pickrunner.mod` file someplace else, just make sure to replace the relative path `..` and `../` to wherever you place the other files.

1.2 Installing Manually

If you don’t want to use the module file, no problem. Just do this:

1. Put the `pickrunner` folder located in “scripts/pickrunner” someplace on the `MAYA_SCRIPT_PATH` - This makes it so that the Pickrunner GUI will load
2. Add the directory to the “userSetup.py” file onto your `MAYA_SCRIPT_PATH`. - This step will override your hotkeys on Maya’s startup
3. Place the “pickrunner_icon.png” in the `XBMLANGPATH` environment variable - Only do this if you want to use the Pickrunner icon(s)
4. Add these commands to a new shelf button

```
from pickrunner import mayarunner  
mayarunner.show()
```

And you’re done.

CHAPTER 2

Scene Setup

If you installed Pickrunner through the “pickrunner.mod” file, you should already have a new shelf called “Pickrunner” on-startup. Just click the Pickrunner GUI button.



This button will load the Pickrunner GUI, which comes in two modes, *Assignment Mode* and *Selection Mode*.

2.1 Assignment Mode

This is the default mode that you’ll see when you open the Pickrunner GUI. Assignment Mode is exactly as it sounds like. It’s the mode that lets you set up Pickrunner-object relationships.

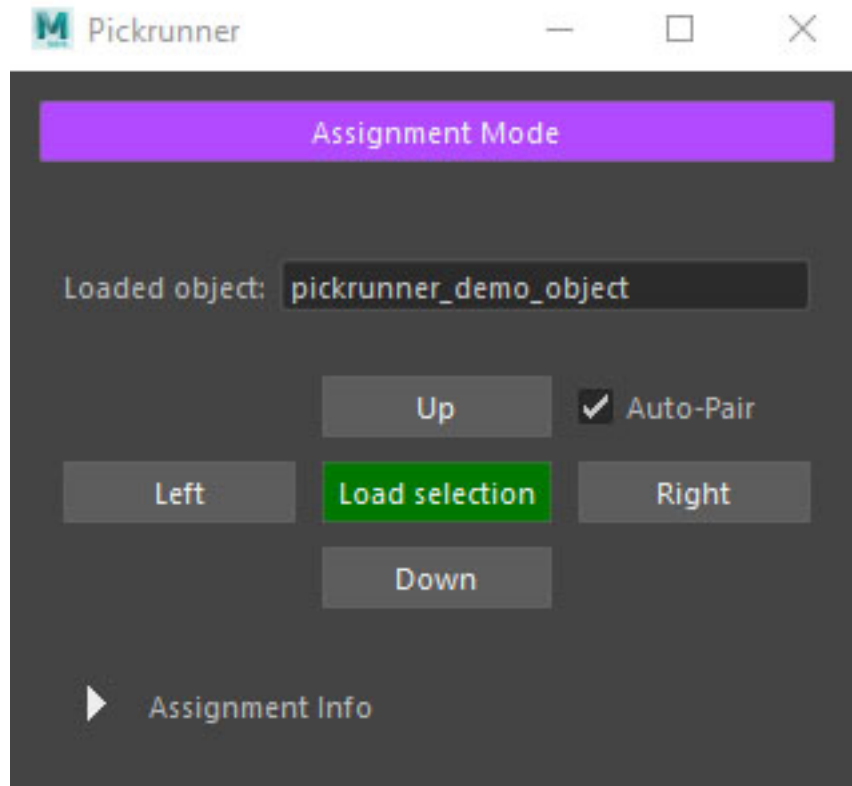
While in Assignment Mode, you should see 4 direction buttons and a button labelled “Load Selection”.

Select an object, for example objectA, and then click “Load Selection”. objectA is now being edited by Pickrunner.

Select another object, for example objectB, and click any of the up, down, left, or right buttons.

Note: If you’ve got “Auto-Pair” enabled, Pickrunner will reflect objects. So when you assign objectB as the “left” direction to objectA, “Auto-Pair” will also set objectA as the “right” direction to objectB. It’s just a timesaver. Turn it off if you don’t want it to do that.

Assuming you’ve done all of the connections you wanted, you’re ready to start using Pickrunner. If you have the direction hotkeys set up correctly, you should be able to press up/down/left/right to move between objects or use Pickrunner’s “Selection Mode”.



2.2 Selection Mode

This is a good mode to test your Pickrunner connections with. Select objects in the scene that have Pickrunner data and press the direction keys. Your selection should move from object to object.

CHAPTER 3

Drawback To Pickrunner

Pickrunner is implemented using node UUIDs, which means you can go from any node to any node, even if the nodes are shading nodes (DG nodes).

The downside to using UUIDs though is that if you export your objects and import them into another scene, Pickrunner won't work. Referencing your nodes will still work though, and in practice that's usually good enough.

If you're looking to contribute, please check out this page, first.

4.1 API Documentation

4.1.1 Setting Up For Development

Make sure to grab the requirements.txt file before trying to build the docs

```
pip -r docs/requirements.txt
```

There are some modules used by Pickrunner that are not mentioned in the requirements.txt file. Namely shiboken, pymel, and maya.cmds. All of these are provided by Maya so we just ignore them for building documentation.

4.1.2 Building The Documentation

Here are the steps to rebuild this documentation:

1. Make sure that the scripts folder is visible in the PYTHONPATH.
2. Run `make html` while cd'ed into the docs folder, or `make.bat` if you're on Windows
3. Alternatively, run `sphinx-build -b html docs/source docs/build`
4. If new Python modules are added, make sure to rebuild their pages, using `docs/regenerate.sh` or `docs/regenerate.bat` if you're on Windows.

4.1.3 Python Documentation

Most (all?) of the documentation is just for the Pickrunner GUI so mileage will vary. The main class of-interest are `pickrunner.gui.BehaviorControl`. For an example of how it's implemented, check out `pickrunner.mayarunner.MayaBehaviorControl`.

pickrunner.gui module

The Pickrunner base interface.

This module contains an abstract controller that is used to implement different DCC environments, such as Maya, and a GUI that the controller can be used for.

class pickrunner.gui.BehaviorControl

Bases: object

An abstract controller that must be implemented in subclasses.

This controller is used to interface with Pickrunner.

classmethod assign(*from_object, direction, to_object, settings=None*)

Set an object to be remapped to another object, given some direction.

Parameters

- **from_object** – The object that will have the direction and to_object stored onto.
- **direction** – Some unique key to store onto from_object. This direction should always point towards to_object. (How direction points to to_object is up to the developer to implement).
- **to_object** – The object to remap to when direction and from_object are given to `BehaviorControl.do_motion()`.

classmethod do_motion(*direction, obj*)

Move the object to a given direction.

How exactly it should “move” must be implemented in subclasses. For example, in Maya, this method will select a node that is associated with the given node-direction pair.

Parameters

- **direction** – The direction to move to.
- **obj** – The object to move from.

classmethod get_object_name(*obj*)

str: Find the unique-name of the given object.

classmethod get_selection()

list: The selected objects in the Maya scene.

classmethod get_settings(*obj*)

dict: Any information stored in the given object that can be used.

pickrunner.mayarunner module

A Maya implementation for Pickrunner.

The user, usually an animator or rigger, can use this GUI to assign object-to-object relationships. Once those relationships are defined, they can use the arrow-keys on their keyboard to move between those objects.

Functionally, this is exactly the same as Maya’s built-in pickWalk command. The difference here however is that pickWalk is notoriously useless because it relies on Maya’s DAG hierarchy, which doesn’t always make navigation easy.

Pickrunner doesn’t care about hierarchy. It can even be used for DG nodes. Take that, pickWalk!

class pickrunner.mayarunner.**MayaBehaviorControl**

Bases: *pickrunner.gui.BehaviorControl*

A controller that implements Maya-specific functions to Pickrunner.

classmethod **assign** (*from_object, direction, to_object, settings=None*)

Set an object to be remapped to another object, given some direction.

Once an object is remapped to another object, we can use that to move Maya's selection around whenever the user asks to.

Parameters

- **from_object** – The object that will have the direction and to_object stored onto.
- **direction** – Some unique key to store onto from_object. This direction should always point towards to_object. (How direction points to to_object is up to the developer to implement).
- **to_object** – The object to remap to when direction and from_object are given to BehaviorControl.do_motion().

classmethod **do_motion** (*direction, obj*)

Change selection to an associated node of obj, given some direction.

Parameters

- **direction** (*str*) – The direction to move to.
- **obj** (*<pm.general.PyNode>*) – The object to get the associated object from.

classmethod **get_object_name** (*obj*)

str: Find the unique-name of the given object.

classmethod **get_selection** ()

list[*<pm.general.PyNode>*]: The selected objects in the Maya scene.

classmethod **get_settings** (*node*)

dict[str]: Get the settings for the given node, if any.

reserved_attribute_name = **'__mayarunner_info'**

pickrunner.mayarunner.do_pickrun_motion (*direction*)

Try to pickrun in a given direction. Otherwise, pickWalk.

Parameters **direction** (*str*) – The direction to walk. Options are: (“up”, “down”, “left”, “right”).

pickrunner.mayarunner.get_uuid (*node*)

str: Get the UUID of the given node, if the node exists.

pickrunner.mui module

Maya-UI functions.

pickrunner.mui.delete_ui_if_exists (**uis*)

Delete UIs using a function wrapper to delete Maya UIs, if they exist.

Parameters ***uis** (*list[str]*) – The UI names to delete.

Returns The wrapped function.

Return type callable

`pickrunner.mui.delete_ui_if_exists_qt (uis)`

Delete UIs using a function wrapper to delete Maya UIs, if they exist.

Parameters `*uis (list[str])` – The UI names to delete.

Returns The wrapped function.

Return type callable

`pickrunner.mui.get_main_menu_bar ()`

<QtWidgets.QMenuBar> or NoneType: The top bar.

`pickrunner.mui.get_main_shelf_name ()`

Get Maya's main shelf widget.

This is just a convenience method for querying `$gShelfTopLevel`.

Example

```
>>> pm.shelfLayout('some_shelf', parent=get_main_shelf_name()).
```

Returns The name/path to the main Maya shelf.

Return type str

`pickrunner.mui.get_main_window ()`

<QtWidgets.QWidget> or NoneType: Cast Maya's window to widget.

`pickrunner.visibility_widget` module

A set of small widgets that are meant to show/hide themselves.

Module contents

p

- `pickrunner`, [12](#)
- `pickrunner.gui`, [10](#)
- `pickrunner.mayarunner`, [10](#)
- `pickrunner.mui`, [11](#)
- `pickrunner.visibility_widget`, [12](#)

A

assign() (pickrunner.gui.BehaviorControl class method), 10

assign() (pickrunner.mayarunner.MayaBehaviorControl class method), 11

B

BehaviorControl (class in pickrunner.gui), 10

D

delete_ui_if_exists() (in module pickrunner.mui), 11

delete_ui_if_exists_qt() (in module pickrunner.mui), 11

do_motion() (pickrunner.gui.BehaviorControl class method), 10

do_motion() (pickrunner.mayarunner.MayaBehaviorControl class method), 11

do_pickrun_motion() (in module pickrunner.mayarunner), 11

G

get_main_menu_bar() (in module pickrunner.mui), 12

get_main_shelf_name() (in module pickrunner.mui), 12

get_main_window() (in module pickrunner.mui), 12

get_object_name() (pickrunner.gui.BehaviorControl class method), 10

get_object_name() (pickrunner.mayarunner.MayaBehaviorControl class method), 11

get_selection() (pickrunner.gui.BehaviorControl class method), 10

get_selection() (pickrunner.mayarunner.MayaBehaviorControl class method), 11

get_settings() (pickrunner.gui.BehaviorControl class method), 10

get_settings() (pickrunner.mayarunner.MayaBehaviorControl class method), 11

get_uuid() (in module pickrunner.mayarunner), 11

M

MayaBehaviorControl (class in pickrunner.mayarunner), 10

P

pickrunner (module), 12

pickrunner.gui (module), 10

pickrunner.mayarunner (module), 10

pickrunner.mui (module), 11

pickrunner.visibility_widget (module), 12

R

reserved_attribute_name (pickrunner.mayarunner.MayaBehaviorControl attribute), 11