nodedge

Release 0.2.0

Anthony De Bortoli

Nov 19, 2022

CONTENTS

Nodedge is a free and open-source software for graphical programming. Try it out and contribute to its development!

Features

- Create your simulation easily thanks to a large variety of customizable built-in blocks.
- Access your models anywhere anytime. Nodedge is free for any purpose.
- Contribute to Nodedge development. The software is based on Python.

Documentation

CHAPTER

ONE

INTRODUCTION

Welcome to the Nodedge User's Manual. This manual contains information on how to use the Nodedge software and includes the documentation for all the modules of the package and examples illustrating their use.

1.1 Installation

The *nodedge* package can be installed using pip or the standard distutils/setuptools mechanisms.

To install using pip:

```
pip install nodedge
```

Alternatively, to use setuptools, first clone or download the source. To install in your default python interpreter, use:

```
git clone git@github.com:nodedge/nodedge.git
cd nodedge
python setup.py install --user
```

1.2 Getting started

To use nodedge simply import the package as follows:

```
>>> import nodedge
```

CHAPTER

TWO

LIBRARY CONVENTIONS

The nodedge library uses a set of standard conventions for the way that different types of standard information used by the library.

2.1 LTI system representation

Linear time invariant (LTI) systems are represented in nodedge in state space, transfer function, or frequency response data (FRD) form. Most functions in the toolbox will operate on any of these data types and functions for converting between compatible types is provided.

2.1.1 State space systems

$$\frac{dx}{dt} = Ax + Bu$$
$$y = Cx + Du$$

where u is the input, y is the output, and x is the state.

2.1.2 Transfer functions

$$G(s) = \frac{\text{num}(s)}{\text{den}(s)} = \frac{a_0 s^m + a_1 s^{m-1} + \cdots + a_m}{b_0 s^n + b_1 s^{n-1} + \cdots + b_n},$$

where n is generally greater than or equal to m (for a proper transfer function).

CHAPTER

THREE

NODEDGE PACKAGE

3.1 Subpackages

3.1.1 nodedge.blocks package

Subpackages

nodedge.blocks.autogen package

Subpackages

nodedge.blocks.autogen.operator package

Submodules

nodedge.blocks.autogen.operator.add_block

nodedge.blocks.autogen.operator.eq_block

nodedge.blocks.autogen.operator.ge_block

nodedge.blocks.autogen.operator.gt_block

nodedge.blocks.autogen.operator.le_block

nodedge.blocks.autogen.operator.lt_block

nodedge.blocks.autogen.operator.mod_block

nodedge.blocks.autogen.operator.mul_block

nodedge.blocks.autogen.operator.pow_block

nodedge.blocks.autogen.operator.sub_block

nodedge.blocks.autogen.operator.truediv_block

Module contents

Module contents

nodedge.blocks.custom package

Submodules

nodedge.blocks.custom.input_block

nodedge.blocks.custom.output_block

Module contents

Submodules

nodedge.blocks.block

nodedge.blocks.block_config

nodedge.blocks.block_exception

Block exception module containing `EvaluationError`, `MissInputError`, and `RedundantInputError` classes.

exception nodedge.blocks.block_exception.**EvaluationError**

Bases: `Exception`

`EvaluationError` class

If a block cannot be evaluated, raise this error.

exception nodedge.blocks.block_exception.**MissInputError**

Bases: *EvaluationError*

`MissInputError` class

If an input is missing to a block, preventing it to be evaluated, raise this error.

exception nodedge.blocks.block_exception.**RedundantInputError**

Bases: *EvaluationError*

`RedundantInputError` class

If two different inputs are connected to a single input socket of a block, raise this error.

3.2 Submodules

Logger module containing function to set up logging functionalities.

nodedge.logger.**setupLogging**(*defaultPath: str = '../logging.yaml'*, *defaultLevel: int = 10*, *envKey: str = 'LOG_CFG'*)

Logging Setup

nodedge.logger.**highLightLoggingSetup**()

3.2.18 nodedge.mdi_area

3.2.19 nodedge.mdi_widget

3.2.20 nodedge.mdi_window

3.2.21 nodedge.node

3.2.22 nodedge.node_tree_widget

3.2.23 nodedge.scene

3.2.24 nodedge.scene_clipboard

3.2.25 nodedge.scene_coder

3.2.26 nodedge.scene_history

3.2.27 nodedge.scene_item_detail_widget

3.2.28 nodedge.scene_items_table_widget

3.2.29 nodedge.serializable

Serializable "interface" module. It is an abstract class.

class nodedge.serializable.**Serializable**

Bases: object

Serializable class

Create data which are common to any serializable object.

It stores the id of the object used in the SceneHistory, the SceneClipboard, and the file structure.

serialize() → OrderedDict

Serialization method to serialize this class data into OrderedDict which can be stored in memory or file easily.

Returns
data serialized in OrderedDict

Return type
OrderedDict

deserialize(*data: dict*, *hashmap: Optional[dict] = None*, *restoreId: bool = True*, **args*, ***kwargs*) → bool

Deserialization method which take data in python `dict` format with helping *hashmap* containing references to existing entities.

Parameters

- **data** (`dict`) – dictionary containing serialized data
- **hashmap** (`dict`) – helper dictionary containing references (by id == key) to existing objects
- **restoreId** (`bool`) – `True` if we are creating new sockets. `False` is useful when loading existing sockets which we want to keep the existing object's *id*

Returns

`True` if deserialization was successful, `False` otherwise

Return type

`bool`

3.2.30 nodedge.socket_type

<ModuleName> module containing ~`nodedge.` class.

class nodedge.socket_type.**MyClass**

Bases: `object`

~`nodedge.` class .

class nodedge.socket_type.**SocketType**(*value*)

Bases: `Enum`

An enumeration.

Any = 0

Number = 1

String = 2

3.2.31 nodedge.utils

3.3 Module contents

CHAPTER

FOUR

EXAMPLES

The source code for the examples below are available in the *examples/* subdirectory of the source code distribution. They can also be accessed online via the nodedge GitHub repository.

4.1 Python scripts

CHAPTER

FIVE

EVALUATION

The evaluation system uses `eval()` and `evalChildren()`. `eval` is supposed to be overridden by your own implementation. The evaluation logic uses Flags for marking the *Nodes* as *Dirty* and/or *Invalid*.

5.1 Evaluation Functions

There are 2 main methods used for the evaluation:

- `eval()`
- `evalChildren()`

These functions are mutually exclusive. That means that `evalChildren` does **not** eval the current *Node*, but only children of the current *Node*.

By default the implementation of `eval()` is "empty" and return 0. However, if successful, eval resets the *Node* not to be *Dirty* nor *Invalid*. This method is supposed to be overridden by your own implementation. If you look for examples, please check out `examples/example_calculator` to get inspiration on how to setup your own *Node* evaluation.

The evaluation takes advantage of the *Node* flags described below.

5.2 `Node` Flags

Each `Node` has 2 flags:

- `Dirty`
- `Invalid`

The *Invalid* flag has always higher priority over *Dirty*. This means that if the *Node* is *Invalid* it does not matter whether it is *Dirty* or not.

To mark a node as *Dirty* or *Invalid* there are respective methods `markDirty()` and `markInvalid()`. Both methods take *bool* parameter for the new state. You can mark *Node* dirty by setting the parameter to `True`. Also you can un-mark the state by passing `False` value.

For both flags there are 3 methods available:

- `markInvalid()` - to mark only the *Node*
- `markChildrenInvalid()` - to mark only the direct (first level) children of the *Node*
- `markDescendantsInvalid()` - to mark it self and all descendant children of the *Node*

The same applies to the *Dirty* flag:

- `markDirty()` - to mark only the *Node*
- `markChildrenDirty()` - to mark only the direct (first level) children of the *Node*
- `markDescendantsDirty()` - to mark it self and all descendant children of the *Node*

Descendants or Children are always connected to the Output(s) of the current *Node*.

When a node is marked as *Dirty* or *Invalid* one of the two event methods, `onMarkedInvalid()` or `onMarkedDirty()`, is called. By default, these methods do nothing. However, they are implemented for you to override and use them in your own evaluation system.

CHAPTER

SIX

EVENT SYSTEM

Nodedge uses its own events (and tries to avoid using `Signal`) to handle logic happening inside the Scene. If a class does handle some events, they are usually described at the top of the page in this documentation.

Any of the events is subscribable to and the methods for registering callback are called:

```
add<EventName>Listener(callback)
```

You can register to any of these events any time.

6.1 Events used in Nodedge:

6.1.1 Scene

Has Been Modified
when something has changed in the *Scene*

Item Selected
when *Node* or *Edge* is selected

Items Deselected
when deselect everything appears

Drag Enter
when something is Dragged onto the *Scene*. Here we do allow or deny the drag

Drop
when we Drop something into the *Scene*

6.1.2 SceneHistory

History Modified
after *History Stamp* has been stored or restored

History Stored
after *History Stamp* has been stored

History Restored
after *History Stamp* has been restored

CHAPTER

SEVEN

SERIALIZATION

All of serializable classes derive from *Serializable* class. *Serializable* does create commonly used parameters for our classes. In our case it is just `id` attribute.

Serializable defines two methods which should be overridden in child classes:

- *serialize()*
- *deserialize()*

According to coding-standards we keep these two functions on the bottom of the class source code.

To contain all of the data we use `OrderedDict` instead of regular *dict*. Mainly because we want to retain the order of parameters serialized in files.

Classes which derive from `Serializable`:

- `Scene`
- `Node`
- `QDMNodeContentWidget`
- `Edge`
- `Socket`
- genindex

Development

You can check out the latest version of the source code with the command:

```
git clone https://github.com/nodedge/nodedge.git
```

You can run a set of unit tests to make sure that everything is working correctly. After installation, run:

```
python setup.py test
```

Your contributions are welcome! Simply fork the GitHub repository and send a pull request.

Links

- Issue tracker: https://github.com/nodedge/nodedge/issues

PYTHON MODULE INDEX

n

INDEX