Hexes Documentation

Release 0.4.0

Kit La Touche

August 08, 2015

Contents

Contents:

CHAPTER 1

Hexes

Service	Status
PyPI	
Downloads	
Waffle.io	
CircleCI	
Read the Docs	

Curses for humans.

This is free software, under a BSD license.

This is very very alpha! I'm working on it, though, and would love your feedback, pull requests, and enthusiasm.

Thanks!

1.1 Features

- It can draw boxes, lay them out, and resize them. FANCY!
- It can put text in boxes and only reveal the appropriate bits of it. SHINY!
- It has an event loop, and allows you to bind to startup and keypresses. SPOOKY!
- It can accept text input and process it. HEAVY!

CHAPTER 2

Installation

At the command line:

```
$ easy_install hexes
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv hexes
$ pip install hexes
```

Really, please don't use `easy_install`.

CHAPTER 3

Usage

To use Hexes in a project:

```
#!/usr/bin/env python

# The basic imports:
from hexes import (
    Application,
    Box,
    Style,
)
from hexes.behaviors import quit
import logging

logging.basicConfig(
    filename='hexes.log',
    level=logging.DEBUG,
)

# We're going to use this in the logic below; not part of Hexes.
import asyncio

# Layout
#
# You can nest boxes indefinitely, though some layouts may fail on some screen
# sizes. You can specify text for boxes, whether that text should be flowed or
# treated as fixed, whether child boxes should be laid out horizontally or
# vertically, height and width for boxes, etc.
ls_box = Box(
    style=Style(
        flow=False,
    ),
)
input_box = Box(
    editable=True,
    style=Style(
        height=3,
    ),
)

root = Box(
    style=Style(
        layout=Style.Layout.Horizontal,
    ),
```

```
    children=(
        Box(
            children=(
                ls_box,
                input_box,
            ),
        ),
        Box(
            style=Style(
                width=20,
            ),
        ),
    ),
)

# Logic
#
# Instantiate the application with the layout attached.
# Register any pre-defined behaviors you want (right now, that's only `quit`)
# using the same mechanism as custom behaviors, `app.on`.
app = Application(root=root)
app.on('q', quit)


# Define custom behavior with the `@app.on` decorator. This decorator
# requires an event identifier, which is either 'ready' or a key identifier
# as returned by `curses.window.getkey`
@app.on('ready')
def input_text(app):
    app.edit(input_box, callback=handle_edit)


@asyncio.coroutine
def handle_edit(app, textbox, characters):
    if ls_box.text is None:
        ls_box.text = ""
    ls_box.text += characters + "\n"
    app.schedule(input_text)


@app.on('j')
def scroll_down(app):
    ls_box.scroll(1)


@app.on('k')
def scroll_up(app):
    ls_box.scroll(-1)

# Run
#
# The context manager helps us clean up no matter what exceptional exit
# conditions we have.
with app:
    app.run()
```

The text input area is still larval. Give me bug reports!

3.1 Musings on the future

What sorts of widgets are important in a terminal app?

- Text areas
- Scrollable text areas
- Auto-scrolling text areas (as for chat or Twitter feed)
- Text input areas

These widgets should be relatively smart, knowing their own dimensions, when to resize, how to listen to things (some sort of data-binding model here?), how to style themselves, etc.

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/wlonk/hexes/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

4.1.4 Write Documentation

Hexes could always use more documentation, whether as part of the official Hexes docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/wlonk/hexes/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *hexes* for local development.

1. Fork the *hexes* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/hexes.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv hexes --python=$(which python3)
$ cd hexes/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass `flake8` and the tests, including testing other Python versions with tox:

```
$ tox -e linting
```

   To get `flake8` and `tox`, just `pip install` them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in `README.rst`.
3. The pull request should work for Python 3.4. Check the tests on your pull request and make sure that the tests pass for all supported Python versions.
4. Please include a cute animal picture with your pull request! Everyone will have a happier time if you do.

4.4 Tips

To run a subset of tests:

```
$ python setup.py test path/to/particular/test
```

CHAPTER 5

Releases

First, get everything you want into master.

This should include changes to `README.rst` and `HISTORY.rst`!

Then, change the version in `hexes/__init__.py`. Commit that, push it.

Make a release on GitHub, generating the tag. Our tags begin with `v`.

Run `make release`.

Dance!

CHAPTER 6

Credits

6.1 Development Lead

- @wlonk: Kit La Touche <kit@transneptune.net>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

CHAPTER 8

0.4.0 (2015-08-08)

- Added support for text input areas.

CHAPTER 9

0.3.1 (2015-05-08)

- Slight unifications to the way you register behaviors.

0.3.0 (2015-05-08)

- Event loop and binding listeners.

CHAPTER 11

0.2.0 (2015-04-26)

- Rendering text into boxes.

CHAPTER 12

0.1.0 (2015-04-25)

- First release on PyPI.

CHAPTER 13

Indices and tables

- genindex
- modindex
- search