rui Documentation

Release 0.8.0

Timothy Hahn

Mar 30, 2017

Contents

Contents:

CHAPTER 1

rui ()

An imperfect Python ECS

- Documentation: http://rui.rtfd.org.
- Github: http://github.com/timothyhahn/rui
- Free software: BSD license
- PyPI: https://pypi.python.org/pypi/rui/

Features

- A simple Python Entity Component System
- Unoptimized
- Favors World based access, eschews Managers (Artemis, which inspired this, uses both)
- **To learn more about Entity Component Systems**
  - http://www.gamedev.net/page/resources/_/technical/game-programming/understanding-component-entity-systems-r3013
  - http://www.randygaul.net/2013/05/20/component-based-engine-design/
  - http://gamadu.com/artemis/manual.html

Examples

- Press Space To Jump: https://github.com/timothyhahn/press-space-to-jump

CHAPTER 2

Installation

At the command line:

```
$ easy_install rui
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv rui
$ pip install rui
```

CHAPTER 3

Usage

```
## Import rui
from rui.rui import Component, System, World

## Define Components
class Position(Component):
    def __init__(self, x, y):
        self.x = x
        self.y = y

class Velocity(Component):
    def __init__(self, x, y):
        self.x = x
        self.y = y

## Define System
class MovementSystem(System):
    def process(self, delta): ## This method is a minimal requirement
        entities = self.world.get_entities_by_components(Position, Velocity)

        for entity in entities:
            position = entity.get_component(Position)
            velocity = entity.get_component(Velocity)
            position.x += velocity.x * delta
            position.y += velocity.y * delta

## Create the world and set up Entities
world = World()

player = world.create_entity(tag='PLAYER') # This does not automatically add the␣
↪entity to the world
                                # You could also do player = Entity('PLAYER')
                                # tag is completely optional, but it allows you to␣
↪look up this entity later
player.add_component(Position(0,0))
player.add_component(Velocity(0,0))
```

```
world.add_entity(player)

world.add_system(MovementSystem())

while True:
    ## Get Player Inputs
    player_by_tag = world.get_entity_by_tag('PLAYER') ## Get the entity by its tag
    world.process() ## The world will step through its motions
```

CHAPTER 4

rui API

rui.rui module

class rui.rui.**Component**
Bases: object

class rui.rui.**Entity**(*tag=''*)
Bases: object

Instances of Entity are unique IDs that hold Components. (optionally) tag is a string that refers to the entity

add_component(**args*, ***kwargs*)
Checks if alive before doing something

check_alive(*function*)

get_component(**args*, ***kwargs*)
Checks if alive before doing something

get_components(**args*, ***kwargs*)
Checks if alive before doing something

get_tag(**args*, ***kwargs*)
Checks if alive before doing something

get_uuid(**args*, ***kwargs*)
Checks if alive before doing something

kill(**args*, ***kwargs*)
Checks if alive before doing something

set_tag(**args*, ***kwargs*)
Checks if alive before doing something

set_world(**args*, ***kwargs*)
Checks if alive before doing something

class rui.rui.**System**
Bases: object

process(*delta*)
Update the system

set_world(*world*)
Sets the world this system belongs to

class rui.rui.**World**(*delta=1*)
Bases: object

A World holds all entities, groups, and systems

add_entities(**entities*)
Add multiple entities to world All members of entities are of type Entity

add_entity(*entity*, *second=False*)
Add entity to world. entity is of type Entity

add_system(*system*)
Add system to the world. All systems will be processed on World.process() system is of type System

create_entity(*tag=''*)
Creates Entity (optionally) tag is a string that is the tag of the Entity.

deregister_entity_from_group(*entity*, *group*)
Removes entity from group

get_delta()
Returns delta

get_entities()
Gets all entities

get_entities_by_components(**components*)
Get entity by list of components All members of components must be of type Component

get_entity_by_tag(*tag*)
Get entity by tag tag is a string that is the tag of the Entity.

get_group(*group*)
Gets a specific group group is the string of a Group

process()
Processes entire world and all systems in it

register_entity_to_group(*entity*, *group*)
Add entity to a group. If group does not exist, entity will be added as first member entity is of type Entity group is a string that is the name of the group

remove_entity(*entity*, *second=False*)
Removes entity from world and kills entity

remove_system(*system*)
Removes system from world and kills system

set_delta(*delta*)
Sets delta

rui.exceptions module

exception rui.exceptions.**DeadEntityError**
Bases: exceptions.Exception

exception rui.exceptions.**DuplicateEntityError**(*entity*)
Bases: exceptions.Exception

exception rui.exceptions.**DuplicateSystemError**(*system*)
Bases: exceptions.Exception

exception rui.exceptions.**NonUniqueTagError**(*tag*)
Bases: exceptions.Exception

exception rui.exceptions.**UnmanagedEntityError**(*entity*)
Bases: exceptions.Exception

exception rui.exceptions.**UnmanagedSystemError**(*system*)
Bases: exceptions.Exception

CHAPTER 5

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:

Types of Contributions

Report Bugs

Report bugs at https://github.com/timothyhahn/rui/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.

Fix Bugs

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

Implement Features

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

Write Documentation

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

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/timothyhahn/rui/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 :)

Get Started!

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

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

```
$ git clone git@github.com:your_name_here/rui.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 rui
$ cd rui/
$ 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:

```
$ flake8 rui tests
$ python setup.py test
$ tox
```

   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.

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 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/timothyhahn/rui/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ python -m unittest tests.test_rui
```

CHAPTER 6

Credits

Development Lead

- Timothy Hahn <timyhahn@gmail.com>

Contributors

None yet. Why not be the first?

CHAPTER 7

History

0.8.0(2014-1-27)

- Added example to README
- Works decently well now!

0.3.8(2014-1-27)

- You can now remove entities from groups

0.3.5(2014-1-27)

- Groups now return empty list if there is no group of that name

0.3.4(2014-1-26)

- Updated usage documentation to fix typo
- Added partial (subset) component selection

0.3.3(2014-1-26)

- Updated README with GitHub and PyPI links
- Systems can now be removed
- Entities can now be removed

- Entities will now check if they are alive or else raise an exception

0.3.2(2014-1-26)

- Added API documentation

0.3.1(2014-1-25)

- Realized I already had an installation and usage section

0.3.0(2014-1-25)

- Added usage documentation
- Fixed capitalization in tests

0.2.2(2014-1-25)

- Syntax error!

0.2.1 (2014-1-25)

- Realized I was even doing that wrong - looked at Facebook's tornado to figure it out

0.2.0 (2014-1-25)

- Forgot a requirement for Travis-CI

0.1.0 (2014-1-25)

- Added simple entity component system
- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

r

Index