diarc Documentation

Release 0.3.0

Dan Brooks

June 25, 2015

Contents

CHAPTER 1

What is diarc?

Diarc is short for di-graph arcgraph. It is a python package used for describing and drawing a multiple-input, multiple-output, directional graph as a modified arcgraph. Diarc provides some built-in tools, as well as an API that can be used to extend its functionality.

CHAPTER 2

Terminology

Drawing the diagram is done by giving three different kinds of descriptions of the graph: connections, layout, and graphics. Connection descriptions provide the topology of the graph using Logical Objects. Layout describes where pieces of the graph should be drawn with respect to other pieces of the graph using Relative Objects. Together, Logical and Relative Objects are paired to form a Diarc Topology - a data structure which provides a 1-1 mapping that specifies an exact diagram to be drawn. Additionally, by changing only Relative Objects, a single graph can be redrawn many different ways. Finally, Visual Objects created using a graphical toolkit are created to mirror the properties of Relative Objects and draw the graph to the screen.

TODO diagram showing how all these objects reference each other

2.1 Logical (Connection) Objects

Logical objects are used to define the graph, without specifing how it should be drawn.

Vertex A vertex in the graph, connected to other vertices through edges.

Edge A directional edge in the graph. Edges have inputs (sources) and outputs (sinks).

Connection Links a single vertex to a single edge in a single direction (either a *source* or a *sink*). A vertex can have many connections, but is limited to a max of two per edge (one Source and one Sink). An edge can have an unlimited number of connections.

Source A *Connection* between a vertex and an *edge* that designates the edge as leaving the vertex.

Sink A *Connection* between a vertex and an *edge* that designates the edge as entering the vertex.

2.2 Relative (Layout) Objects

Relative objects are used to define how a graph is laid out. These objects are usually created automatically along with Logical Objects, but are not assigned the values that specify their positioning.

Block Corresponds to a Vertex in a 1-1 relationship. Blocks are drawn in a single horizontal line across the middle of the graph. The order in which blocks are drawn is determined by their index number - a unique interger value that must be greater or equal to zero. Blocks are drawn in order of increasing index value from left to right.

Band Correspond to an Edge. This is represented by a line arcing between Snaps inside Blocks. These lines are drawn above and/or below the line of blocks, depending on the direction of the edge. Bands drawn above the blocks are called "positive bands" and represent and edge traveling from from left to right. In other words, the edge's sources are to the left of the edge's sinks. Bands drawn below the blocks are called "negative bands" and represent edges with sources to the right of the sinks. An Edge will always have two Bands (one positive and

one negative). Each Band is drawn such that the arc reaches a certain distance away from the line of blocks. This distance is called the Band's altitude, and is represented by a non-zero unique integer. Positive bands have positive altitudes, while negative bands have negative altitudes. Since bands can overlap each other, they need also need to specify the order in which they are layered. This is done using a bands rank, a positive integer value that must be unique among bands on the same side of the Blocks.

Snap Corresponds to a Connection in a 1-1 relationship. Snaps are split into two groups, Emitters (for Sources) and Collectors (for Sinks). Snaps are drawn inside their Vertex's Block, with Emitters on the left and Collectors on the right. Within these groups, the order of the snaps is determined by the snaps 'order', a positive integer that must be unique among its groups. A snaps absolute location is denoted by a combination of the block it resides in, whether it is in the collectors or emitters, and its order value. This is designated by a unique identifier called a snapkey, denoted as <blockindex><[E]mitter|[C]ollector><snaporder>. For example, the second snap from the left in the collector of block with index 5 is denoted as "5c2".

2.3 Visual (Graphical) Objects

Visual Objects are often specified using a graphical library such as Qt. They specify everything from color and size to actual positioning on the screen.

BlockItem Corresponds to a single _Block (and Vertex).

BandItem Corresponds to a single Band (and Edge).

SnapItem Corresponds to a single Snap (and Connection).

CHAPTER 3

topology - the diarc graph data structure

This module contains the graph data structures used by diarc.

class topology.**Topology**
Storage container for describing the graph and diagram

vertices
An unordered list of all *Vertex* objects.

edges
An unordered list of all *Edge* objects

blocks
A dictionary of *Block* objects indexed by *Block.index*. The dictionary is generated every time it is requested. Only blocks with proper *Block.index* values are included.

bands
A dictionary of *Band* objects, listed by *Band.altitude*. Bands which have not been assigned altitudes are not reported. All bands that have an altitude (regardless of if they are being used (indicated by isUsed) are reported.

snaps
A dictionary of Snap objects, by snapkey. Snaps which have not been assigned an order are not reported. All snaps that have an order regardless of if they are being used (indicated by isUsed) are reported.

hide_disconnected_snaps

class topology.**Vertex**
A Vertex in a directional graph. A vertex can connect to multiple edges as either an input (source) or output (sink) to the edge. It is graphically represented by a Block object.

sources
Unordered list of outgoing connections (*Source* objects)

sinks
Unordered list of incomming connections. (*Sink* objects)

block
Block object for this Vertex. It is created when the vertex is instantiated, and cannot be reassigned.

release()
Removes this vertex from the topology. This additionally removes all its associated *Connection* and *Block* objects from the topology.

class topology.**Edge**
A directional multiple-input multiGple-output edge in the graph. Inputs (sources) and outputs (sinks) are linked to vertices. An edge is represented graphically by either 1 or 2 Band objects.

sources

sinks

posBand

negBand

release()

class topology.**Connection**
A base class for connecting a vertex to an edge, but without specifing the nature of the connection (input or output). Rather then using this class directly, Source or Sink objects should be used.

snap

edge

vertex

block

release()

class topology.**Source**(*Connection*)
A logical connection from a Vertex to an Edge. Graphically represented by a Snap object.

release()

class topology.**Sink**(*Connection*)
A logical connection from an Edge to a Vertex. Graphically represented by a Snap object.

release()

class topology.**Block**
Visual Representation of a Vertex Visual Parameters Index - Unique int value to determine order in which to draw blocks.

Lower values to the left, higher to the right. Indices do not necessarily need to be consecutive.

index
Defines the order in which blocks are arranged. This value is initially unset (defaults to None). For the block to be displayed as part of the graph, this value must be changed to a positive integer that is unique among blocks.

vertex
Returns the logical component (Vertex) for this relative object. The vertex is bound to this block, and cannot be changed.

emitter
Dictionary of Snaps that represent source connections for this block. Only snaps which have been assigned an order value are represented, since the order is used as the dictionary key. If hide_disconnected_snaps is set in the topology, only return snaps where isLinked() is true.

collector
Dictionary of Snap objects that represent sink connections for this block. Only snaps which have been assigned an order value are represented, since the order is used as the dictionary key. If hide_disconnected_snaps is set in the topology, only return snaps where isLinked() is true.

leftBlock
returns the block to the left, determined by block which has the next lowest index value.

rightBlock
returns the block to the right, determined by block which has the next highest index value.

class topology.**Band**

Visual Representation of an Edge. An Edge can have up to two Bands - one with positive altitude and one negative. Visual Parameters Rank - the Z drawing order (higher values closer to user) Altitude - the distance above or below the Block ribbon

altitude

rank

edge

emitters

collectors

isPositive

topBand

bottomBand

isUsed()

---

class diarc.topology.**Vertex**(*topology*)

A Vertex in a directional graph. A vertex can connect to multiple edges as either an input (source) or output (sink) to the edge. It is graphically represented by a Block object.

Sources - outgoing connections to Edges Sinks - incomming connections from Edges

sources

Returns an unordered list of outgoing connections (Source objects) from this vertex.

sinks

Returns an unordered list of outgoing connections (Sink objects) from this vertex.

block

Returns the relative graphical object (Block) for this Vertex. The block cannot be changed

class diarc.topology.**Edge**(*topology*)

A directional multiple-input multiGple-output edge in the graph. Inputs (sources) and outputs (sinks) are linked to vertices. An edge is represented graphically by either 1 or 2 Band objects.

Sources - inputs from vertices Sinks - outputs to vertices

release()

Removes this edge from the topology

sources

returns list of all source connections to this edge

sinks

returns list of all sink connections from this edge

class diarc.topology.**Connection**(*topology*, *vertex*, *edge*)

A base class for connecting a vertex to an edge, but without specifing the nature of the connection (input or output). Rather then using this class directly, Source or Sink objects should be used.

release()

Removes this connection between a vertex and an edge from the topology. This does NOT release either the vertex or the edge objects, it simply removes this particular reference to them.

class diarc.topology.**Source**(*topology*, *vertex*, *edge*)

A logical connection from a Vertex to an Edge. Graphically represented by a Snap object.

class diarc.topology.**Sink**(*topology*, *vertex*, *edge*)
A logical connection from an Edge to a Vertex. Graphically represented by a Snap object.

class diarc.topology.**Block**(*vertex*)
Visual Representation of a Vertex Visual Parameters Index - Unique int value to determine order in which to draw blocks.

Lower values to the left, higher to the right. Indices do not necessarily need to be consecutive.

vertex
Returns the logical component (Vertex) for this relative object. The vertex is bound to this block, and cannot be changed.

emitter
Dictionary of Snaps that represent source connections for this block. Only snaps which have been assigned an order value are represented, since the order is used as the dictionary key. If hide_disconnected_snaps is set in the topology, only return snaps where isLinked() is true.

collector
Dictionary of Snaps that represent sink connections for this block. Only snaps which have been assigned an order value are represented, since the order is used as the dictionary key. If hide_disconnected_snaps is set in the topology, only return snaps where isLinked() is true.

class diarc.topology.**Band**(*edge*, *isPositive*)
Visual Representation of an Edge. An Edge can have up to two Bands - one with positive altitude and one negative. Visual Parameters Rank - the Z drawing order (higher values closer to user) Altitude - the distance above or below the Block ribbon

emitters
returns a list of source snaps that reach this band

collectors
returns list of sink snaps that reach this band

isUsed()
returns true if this band is needed to represent connections on its edge, else false. This is determined by checking if any sources reach this band.

topBand
Returns the band with the next highest altitude, or None if either there is no band above this one or the block ribbon is above it. Bands for which isUsed() is false are skipped over.

bottomBand
Returns the band with the next lowest altitude, or None if either there is no band below this one or the block ribbon is below it. Bands for which isUsed() is false are skipped over.

class diarc.topology.**Snap**(*connection*)
Visual Representation of a Source or Sink. Snaps are layedout horizontally inside of an Emitter or Collector of a Block. A Snap provides a mapping between a Source/Sink and one or two Bands associated with a single Edge. Visual Layout Paramters Order - 0-indexed order in which to draw snaps within an Emitter or Collector

snapkey()
generates the snapkey for this snap

posBandLink
returns the positive band connection - if it exists. Just because a positive band link exists does not mean that it should be drawn. The check for if we should draw the connection happens at drawing time when we decide if we should be using positive or negative

negBandLink
returns the negative band connection - if it exists. See posBand for more details.

isLinked()
returns true if this snap is connected to at least one sink, else false.

isUsed()
returns true if topology.hide_disconnected_snaps is True and isLinked is True, or if topology.hide_disconnected_snaps is false. Otherwise, return true.

leftSnap
Returns the snap directly to the left of this snap within either an emitter or collector. Returns None if this is leftmost snap.

rightSnap
Returns the snap directly to the right of this snap within either an emitter or collector. Returns None if this is rightmost snap.

CHAPTER 4

Building

CHAPTER 5

Building the Documentation

First you need `sphinx` installed, on Ubuntu:

```
$ sudo apt-get install python-sphinx
```

On other platforms use pip:

```
$ sudo pip install Sphinx
```

You have to have built the package first, then you mush source the resulting devel or install space:

```
$ source /path/to/space/setup.bash
```

Then from the capabilities source folder you can build the docs:

```
$ cd docs
$ make html
```

The resulting docs will be generated to `doc/.build/html/index.html`.

CHAPTER 6

Running the Tests

CHAPTER 7

Running the Code

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

d

t

T

V