

Agile Visualization with Roassal

Object Profile
info@objectprofile.com
31/08/2014

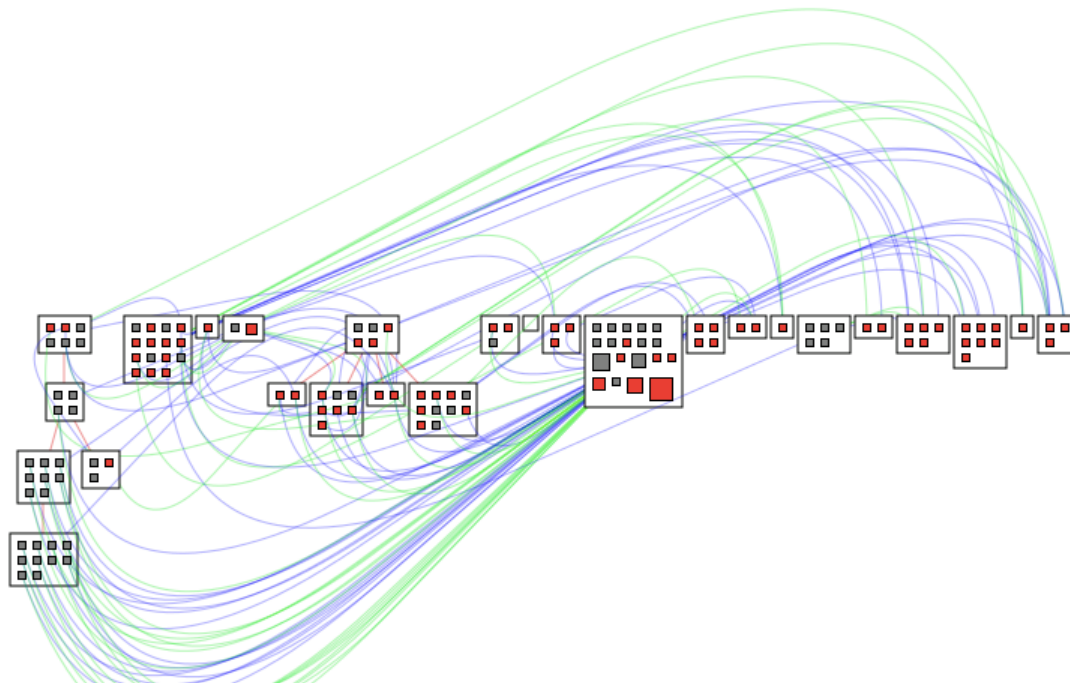


Although often used to visualize software-related data, Roassal visualizes any data

Roassal is used in several industrial and open-source projects

Roassal is distributed under the MIT License

This presentations contains screenshots



Playground

Playground

```
b := RTGraphBuilder new.  
b nodes color: Color gray.  
b layout grid.  
b addAll: (1 to: 50).  
b build
```

a RTGraphBuilder

View

State

Meta

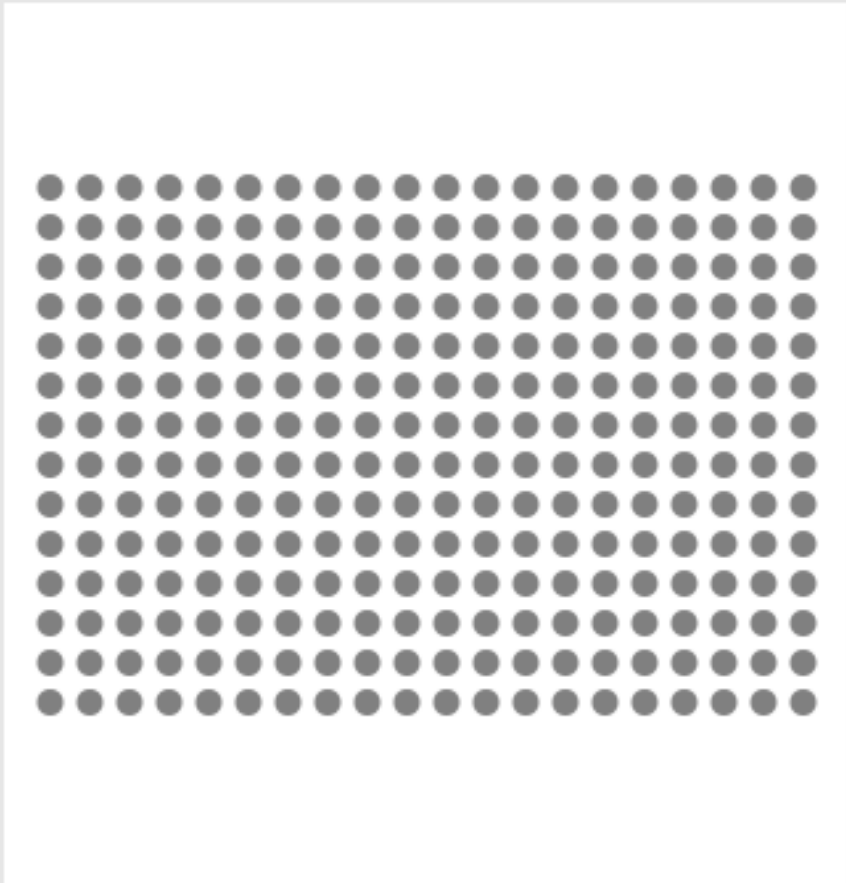


Playground

```
"This code, and the subsequent ones
visualize two large class hierarchies"
b := RTGraphBuilder new.
b nodes color: Color gray.
b layout grid.
b addAll: (RTObject withAllSubclasses,
TRShape withAllSubclasses).
b build
```

a RTGraphBuilder

View State Meta



The image shows a software interface titled "Playground" with two main panels. The left panel contains a code editor with a light blue background, displaying a block of Smalltalk code. The code starts with a comment and then defines a variable 'b' as a new RTGraphBuilder. It sets the node color to gray and the layout to a grid. Finally, it adds all subclasses of RTObject and TRShape to the graph and calls the build method. The right panel, titled "a RTGraphBuilder", has three tabs: "View", "State", and "Meta". The "View" tab is selected, showing a 15x15 grid of 225 gray circular nodes. The interface has a standard window title bar with close, minimize, and maximize buttons, and a status bar at the bottom.

Playground

Playground

```
"This code, and the subsequent ones
visualize two large class hierarchies"
"Classes with 'Shape' in their name
are in red"
b := RTGraphBuilder new.
b nodes
  if: [ :aClass | '*Shape*' match:
aClass name ];
  color: Color red.
b nodes color: Color gray.
b layout grid.

b global
  normalizeSize: #numberOfLinesOfCode
min: 5 max: 30.

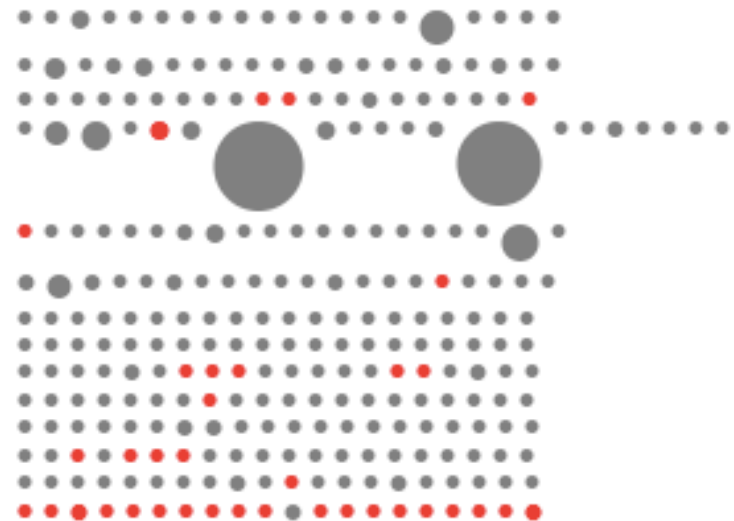
b addAll: (RTObject withAllSubclasses,
TRShape withAllSubclasses).
b build
```

a RTGraphBuilder

View

State

Meta



Playground

Playground

```
"This code, and the subsequent ones
visualize two large class hierarchies"
"Classes with 'Shape' in their name
are in red"
b := RTGraphBuilder new.
b nodes
  if: [ :aClass | '*Shape*' match:
aClass name ];
  color: Color red.
b nodes color: Color gray.
b layout grid.

b global
  normalizeSize: #numberOfLinesOfCode
min: 5 max: 30 using: #sqrt.

b addAll: (RTObject withAllSubclasses,
TRShape withAllSubclasses).
b build
```

a RTGraphBuilder

View

State

Meta



Playground

Playground

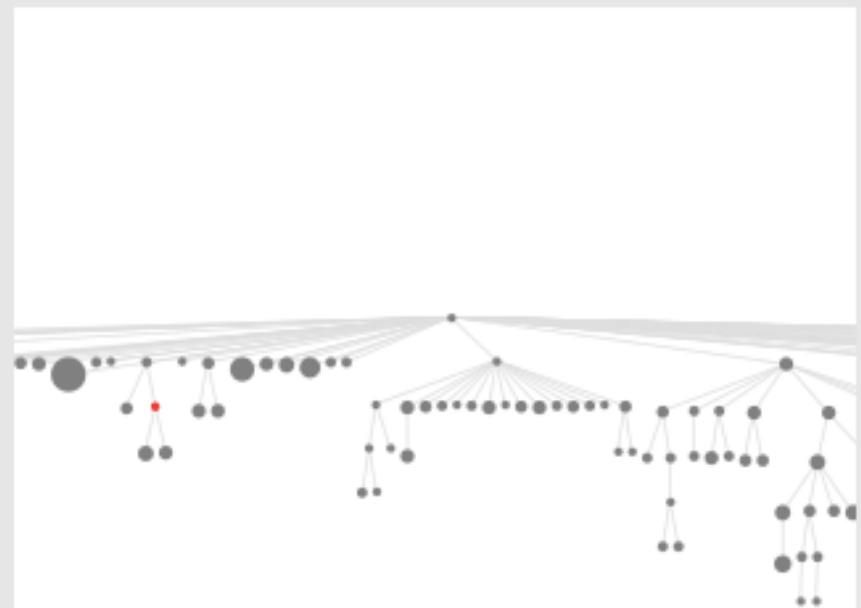
```
b := RTGraphBuilder new.  
b nodes  
  if: [ :aClass | '*Shape*' match:  
aClass name ];  
  color: Color red.  
b nodes color: Color gray.  
b edges connectFrom: #superclass;  
useInLayout.  
b layout tree.  
  
b global  
  normalizeSize: #numberOfLinesOfCode  
min: 5 max: 30 using: #sqrt.  
b addAll: (RTObject withAllSubclasses,  
TRShape withAllSubclasses).  
b build
```

a RTGraphBuilder

View

State

Meta



Playground

Playground

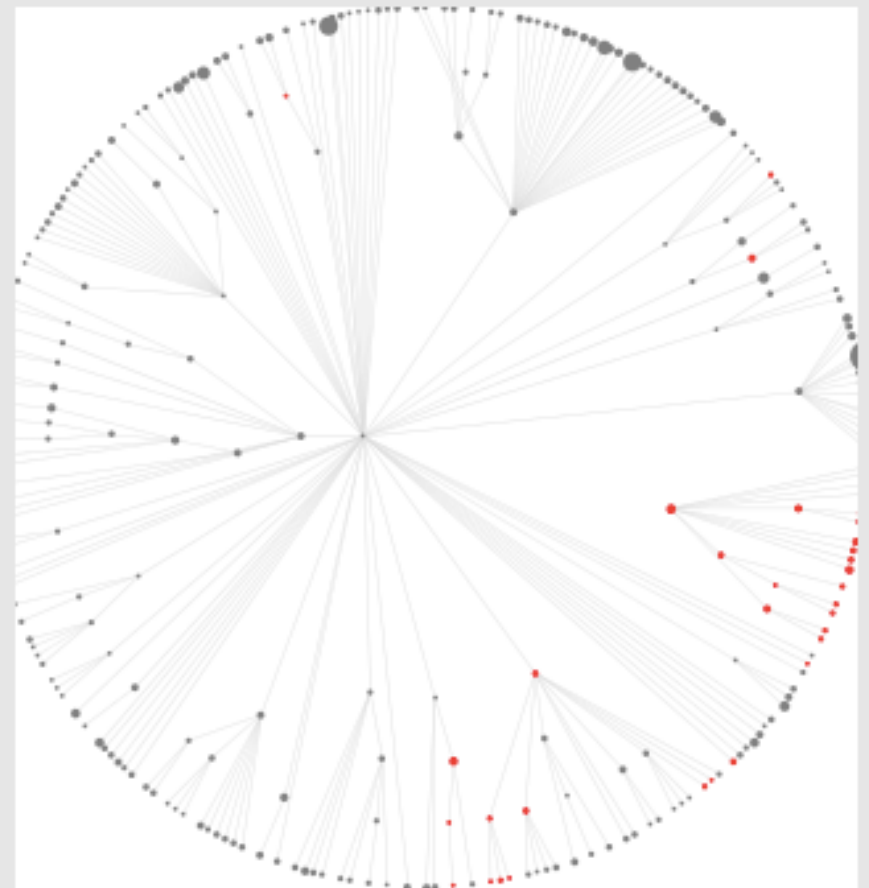
```
b := RTGraphBuilder new.  
b nodes  
  if: [ :aClass | '*Shape*' match:  
aClass name ];  
  color: Color red.  
b nodes color: Color gray.  
b edges connectFrom: #superclass;  
useInLayout.  
b layout cluster.  
  
b global  
  normalizeSize: #numberOfLinesOfCode  
min: 5 max: 30 using: #sqrt.  
b addAll: (RTObject withAllSubclasses,  
TRShape withAllSubclasses).  
b build
```

a RTGraphBuilder

View

State

Meta



Playground

Playground

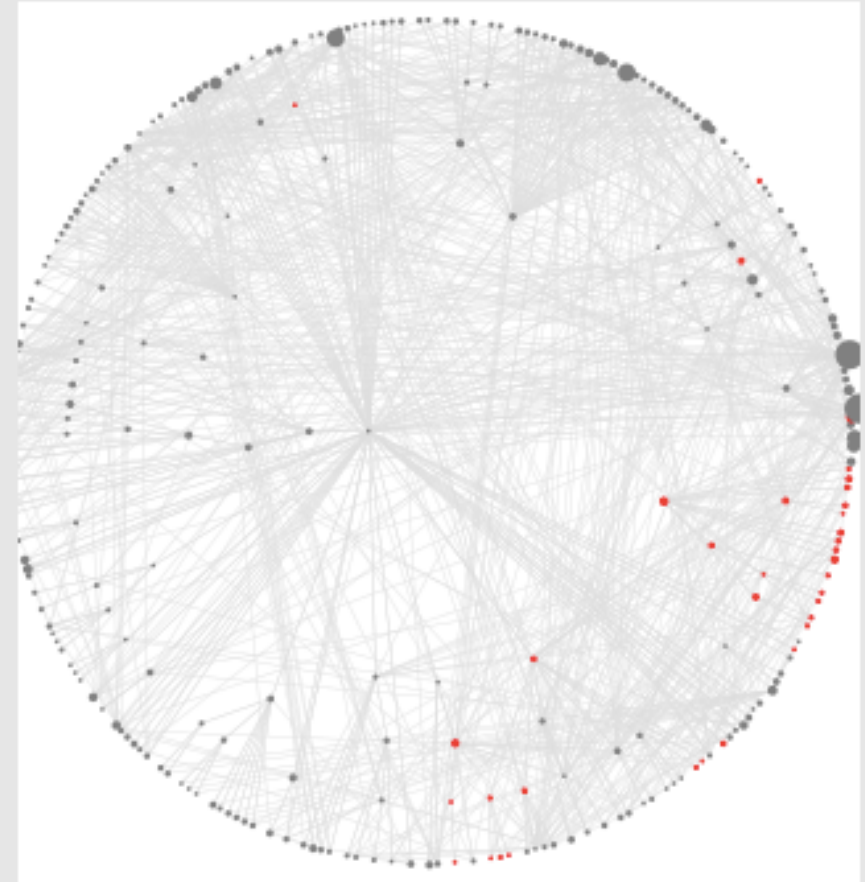
```
b := RTGraphBuilder new.  
b nodes  
  if: [ :aClass | '*Shape*' match:  
aClass name ];  
  color: Color red.  
b nodes color: Color gray.  
b edges connectFrom: #superclass;  
useInLayout.  
b edges connectTo: #dependentClasses.  
b layout cluster.  
  
b global  
  normalizeSize: #numberOfLinesOfCode  
min: 5 max: 30 using: #sqrt.  
b addAll: (RTObject withAllSubclasses,  
TRShape withAllSubclasses).  
b build
```

a RTGraphBuilder

View

State

Meta



Playground

Playground

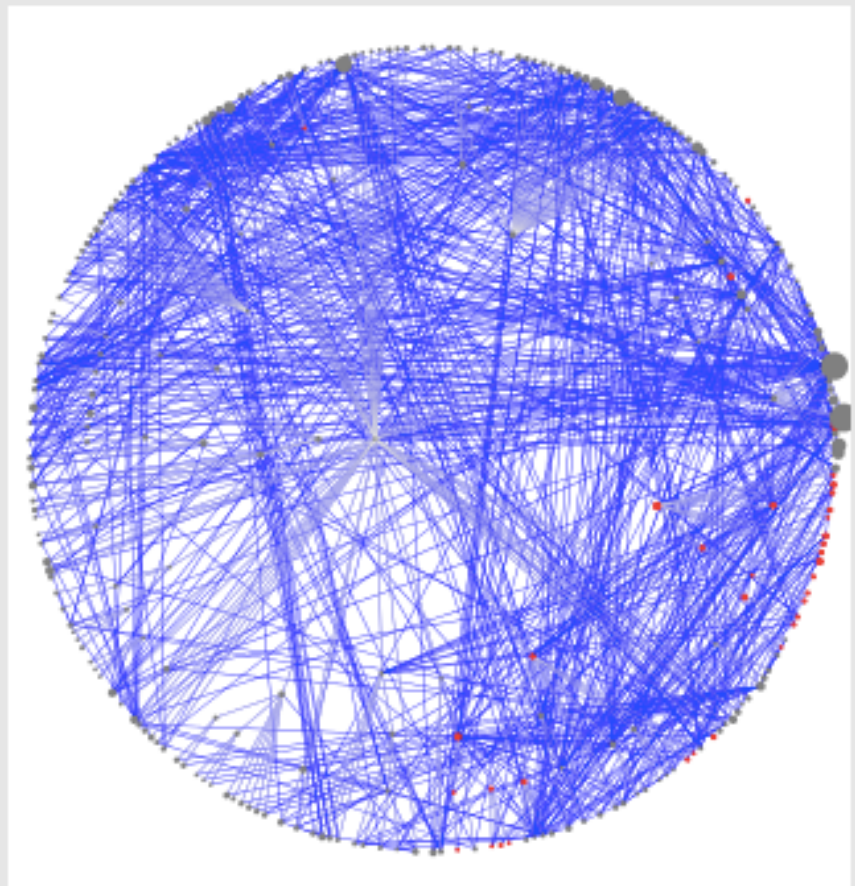
```
b := RTGraphBuilder new.  
b nodes  
  if: [ :aClass | '*Shape*' match:  
aClass name ];  
  color: Color red.  
b nodes color: Color gray.  
b edges connectFrom: #superclass;  
useInLayout.  
b edges  
  connectTo: #dependentClasses;  
  color: Color blue.  
b layout cluster.  
  
b global  
  normalizeSize: #numberOfLinesOfCode  
min: 5 max: 30 using: #sqrt.  
b addAll: (RTObject withAllSubclasses,  
TRShape withAllSubclasses).  
b adjustCamera.
```

a RTGraphBuilder

View

State

Meta



Playground

Playground

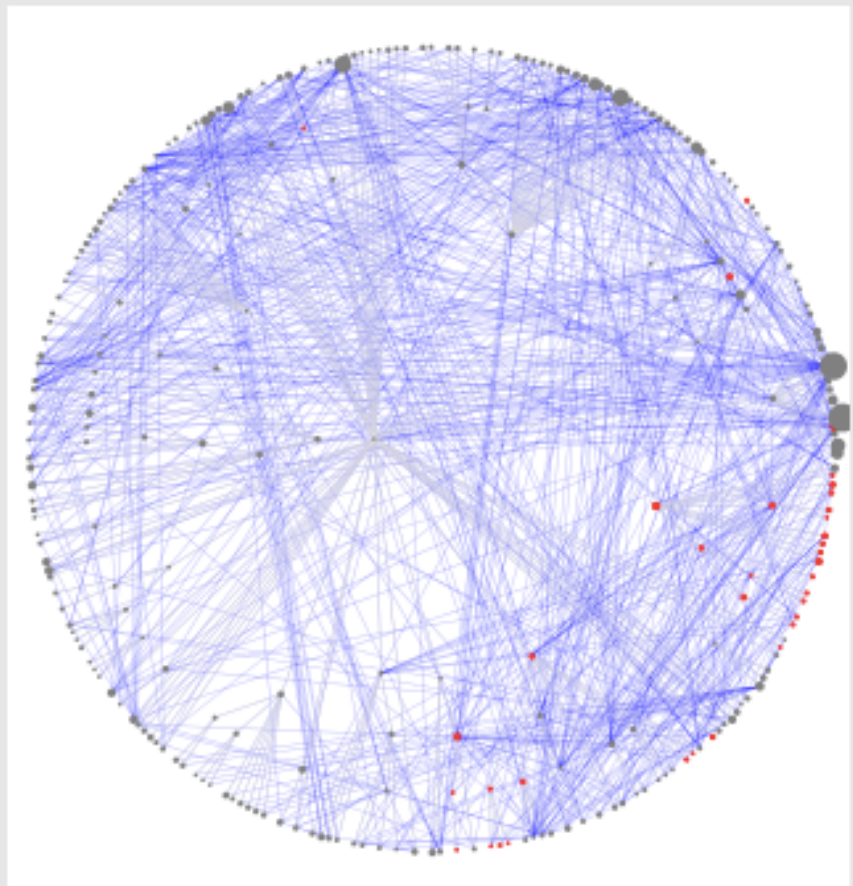
```
b := RTGraphBuilder new.  
b nodes  
  if: [ :aClass | '*Shape*' match:  
aClass name ];  
  color: Color red.  
b nodes color: Color gray.  
b edges connectFrom: #superclass;  
useInLayout.  
b edges  
  connectTo: #dependentClasses;  
  color: (Color blue alpha: 0.3).  
b layout cluster.  
b global  
  normalizeSize: #numberOfLinesOfCode  
min: 5 max: 30 using: #sqrt.  
b addAll: (RTObject withAllSubclasses,  
TRShape withAllSubclasses).  
b adjustCamera.  
b build.
```

a RTGraphBuilder

View

State

Meta



Playground

Playground

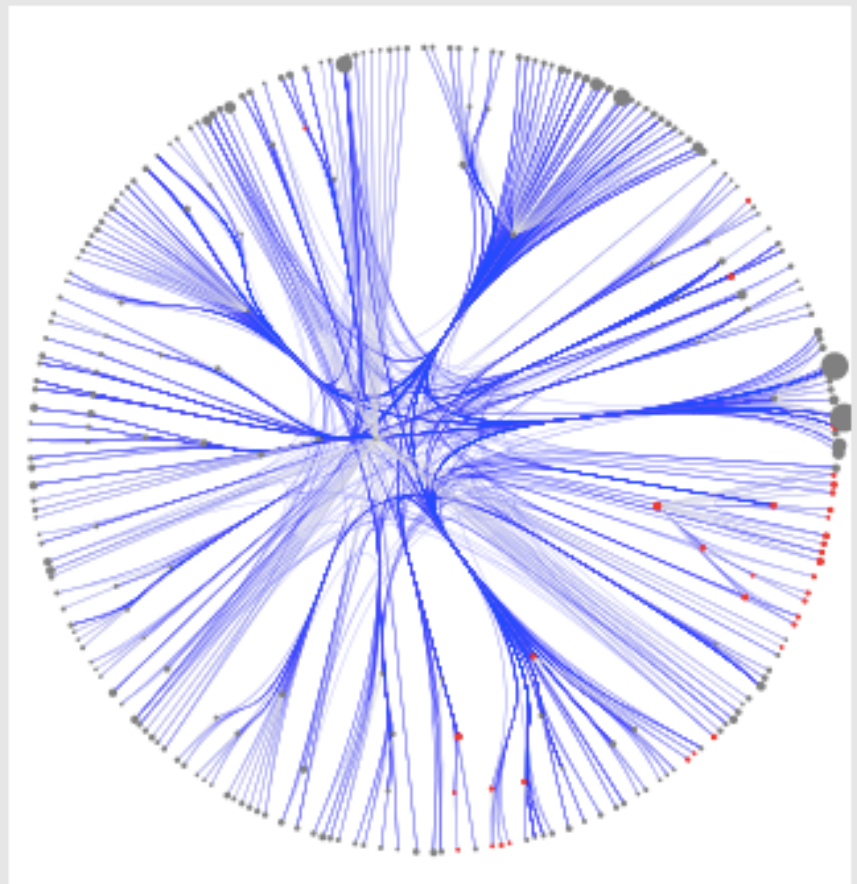
```
b := RTGraphBuilder new.  
b nodes  
  if: [ :aClass | '*Shape*' match:  
aClass name ];  
  color: Color red.  
b nodes color: Color gray.  
b edges connectFrom: #superclass;  
useInLayout.  
b edges  
  connectTo: #dependentClasses;  
  follow: #superclass;  
  color: (Color blue alpha: 0.3).  
b layout cluster.  
b global  
  normalizeSize: #numberOfLinesOfCode  
min: 5 max: 30 using: #sqrt.  
b addAll: (RTObject withAllSubclasses,  
TRShape withAllSubclasses).  
b adjustCamera.
```

a RTGraphBuilder

View

State

Meta



Playground

Playground

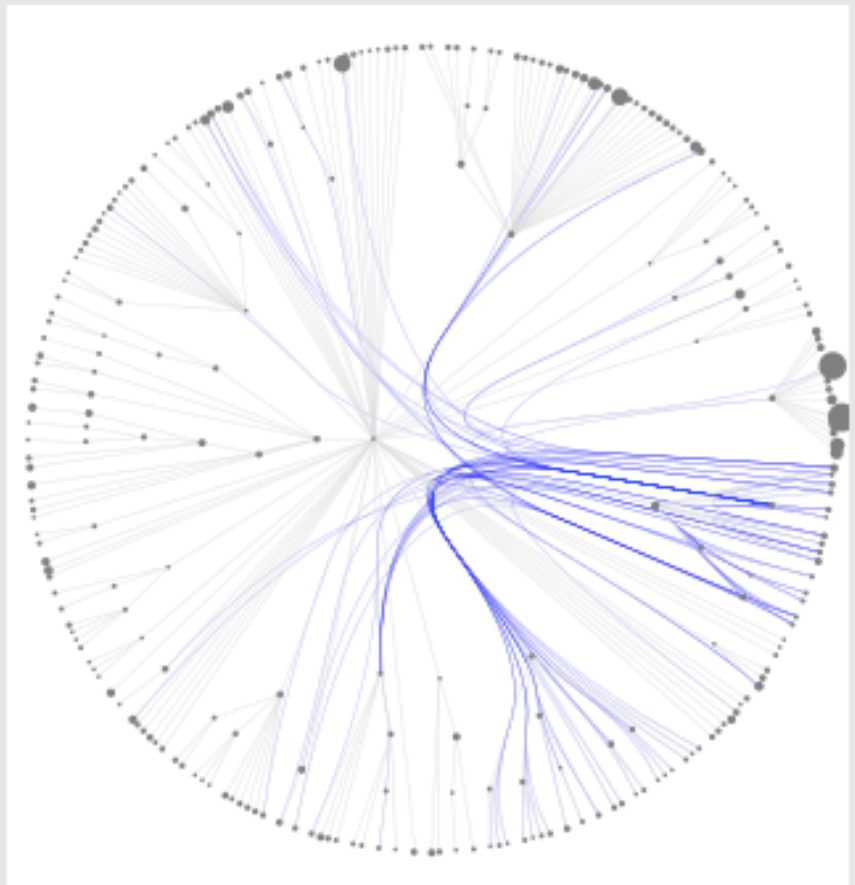
```
b := RTGraphBuilder new.  
b nodes color: Color gray.  
b edges connectFrom: #superclass;  
useInLayout.  
b edges  
  if: [ :fromClass :toClass | toClass  
inheritsFrom: TRShape ];  
  connectTo: #dependentClasses;  
  follow: #superclass;  
  color: (Color blue alpha: 0.3).  
b layout cluster.  
b global  
  normalizeSize: #numberOfLinesOfCode  
min: 5 max: 30 using: #sqrt.  
b addAll: (RTObject withAllSubclasses,  
TRShape withAllSubclasses).  
b adjustCamera.  
b build.
```

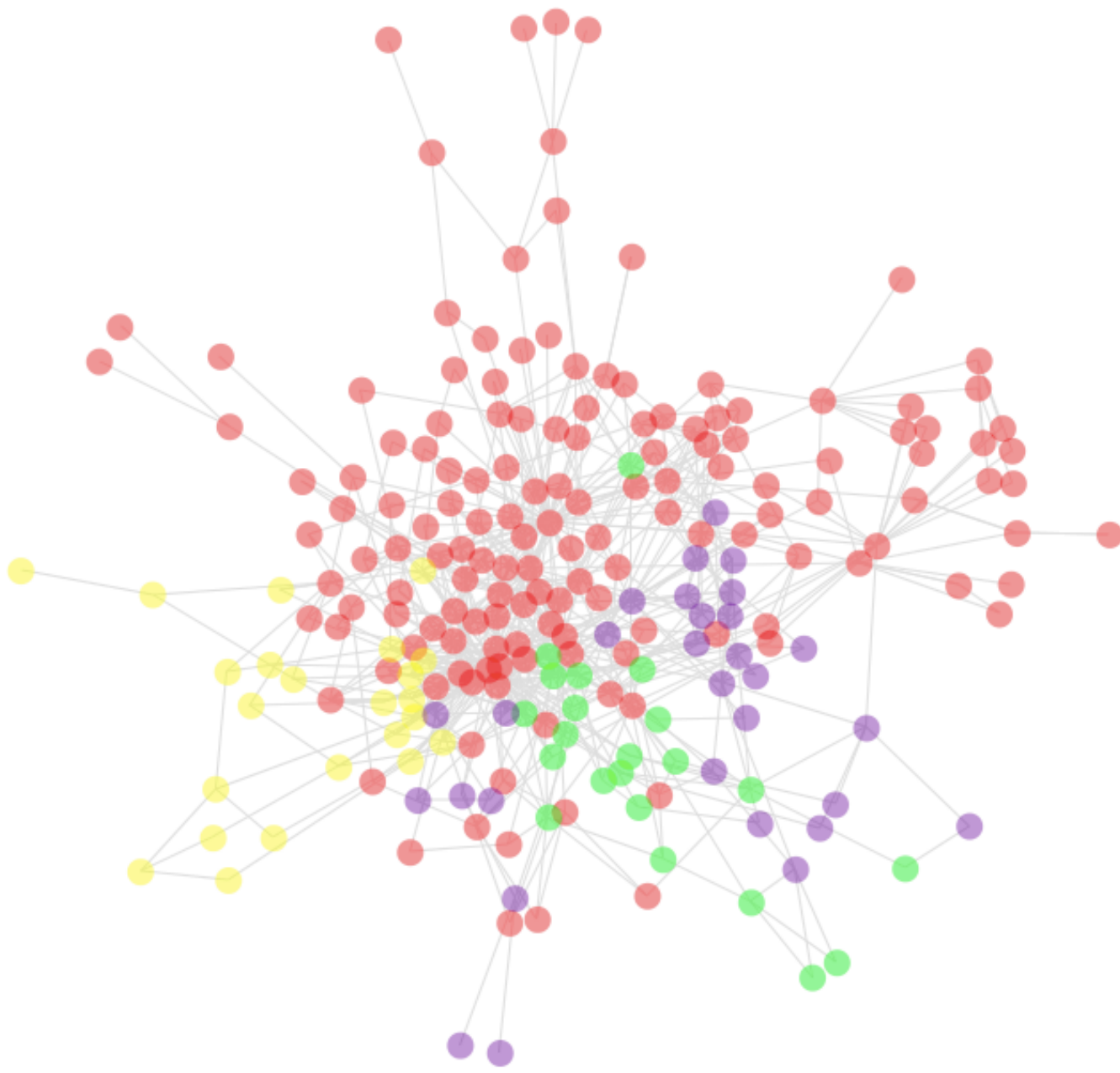
a RTGraphBuilder

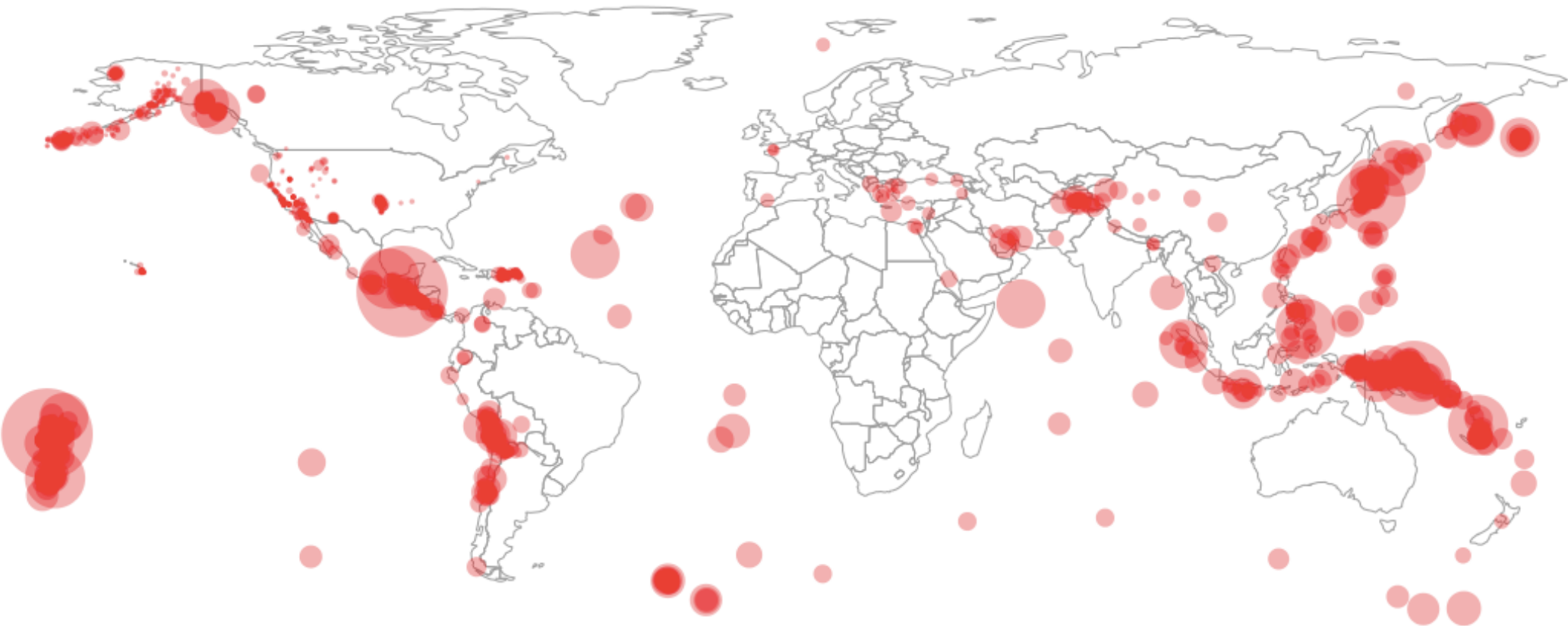
View

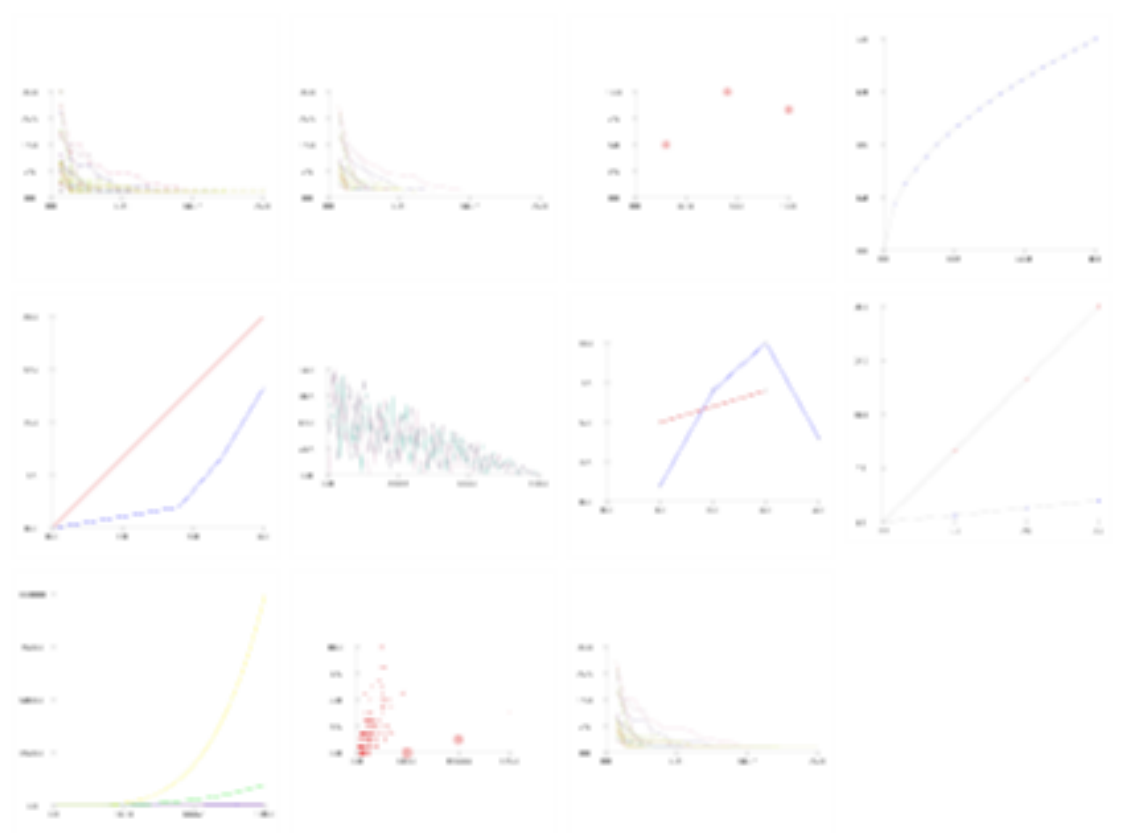
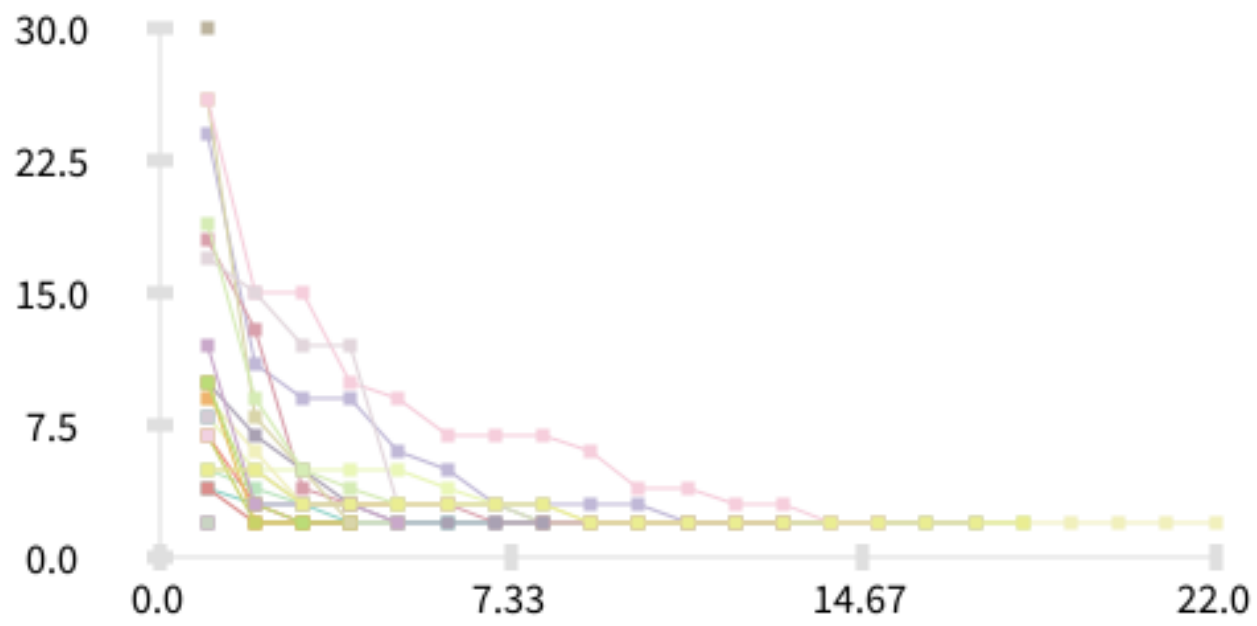
State

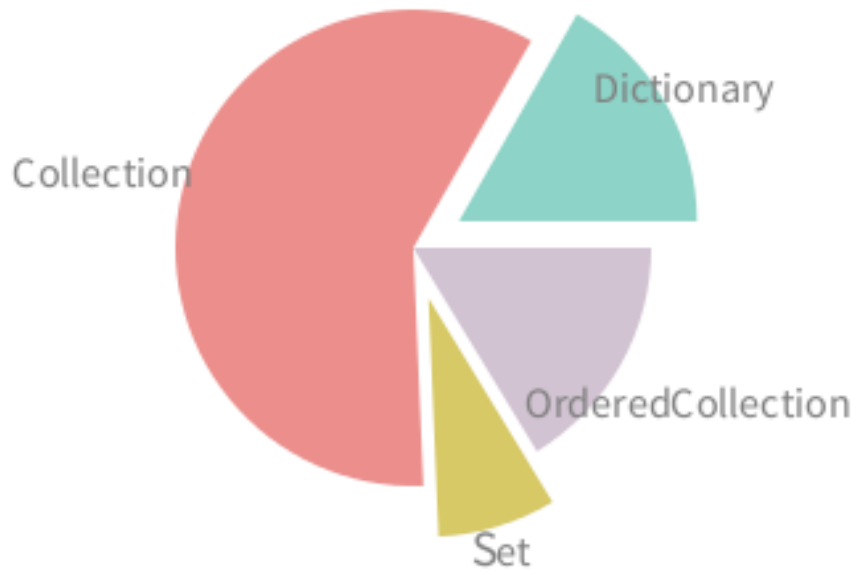
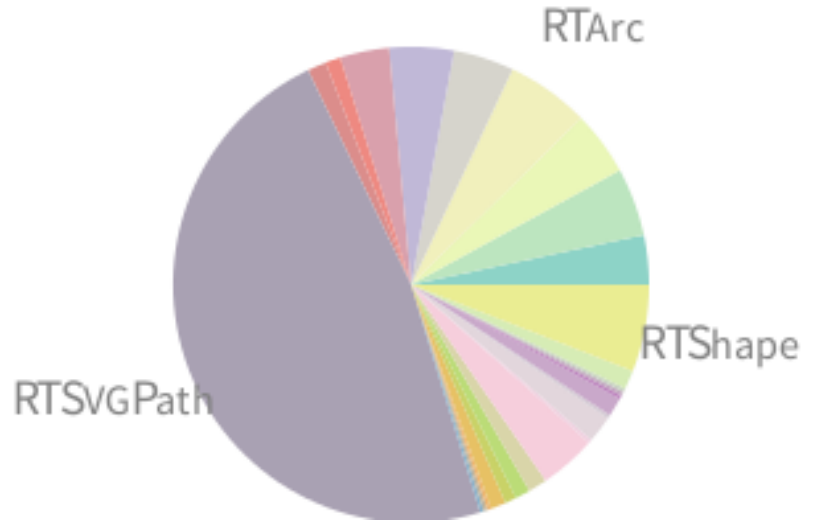
Meta











Playground

Playground



a RTView (a RTView)



View

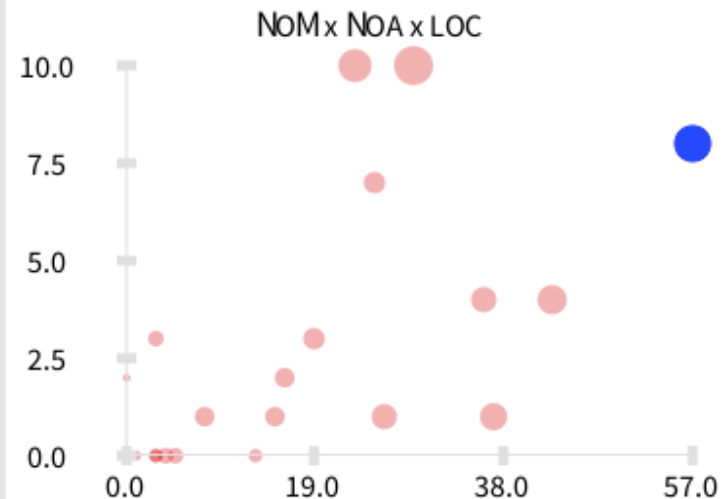
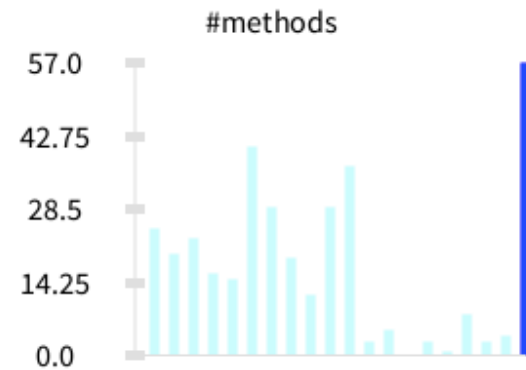
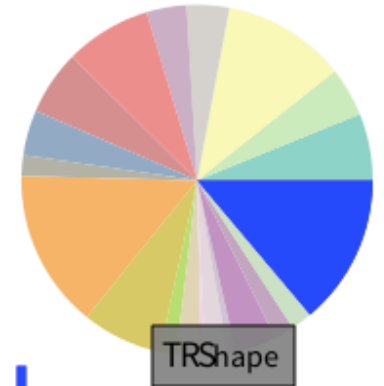
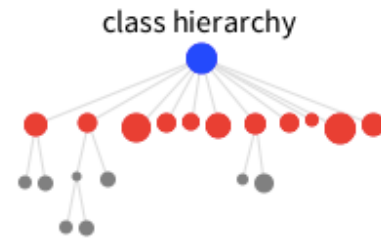
State

Elements

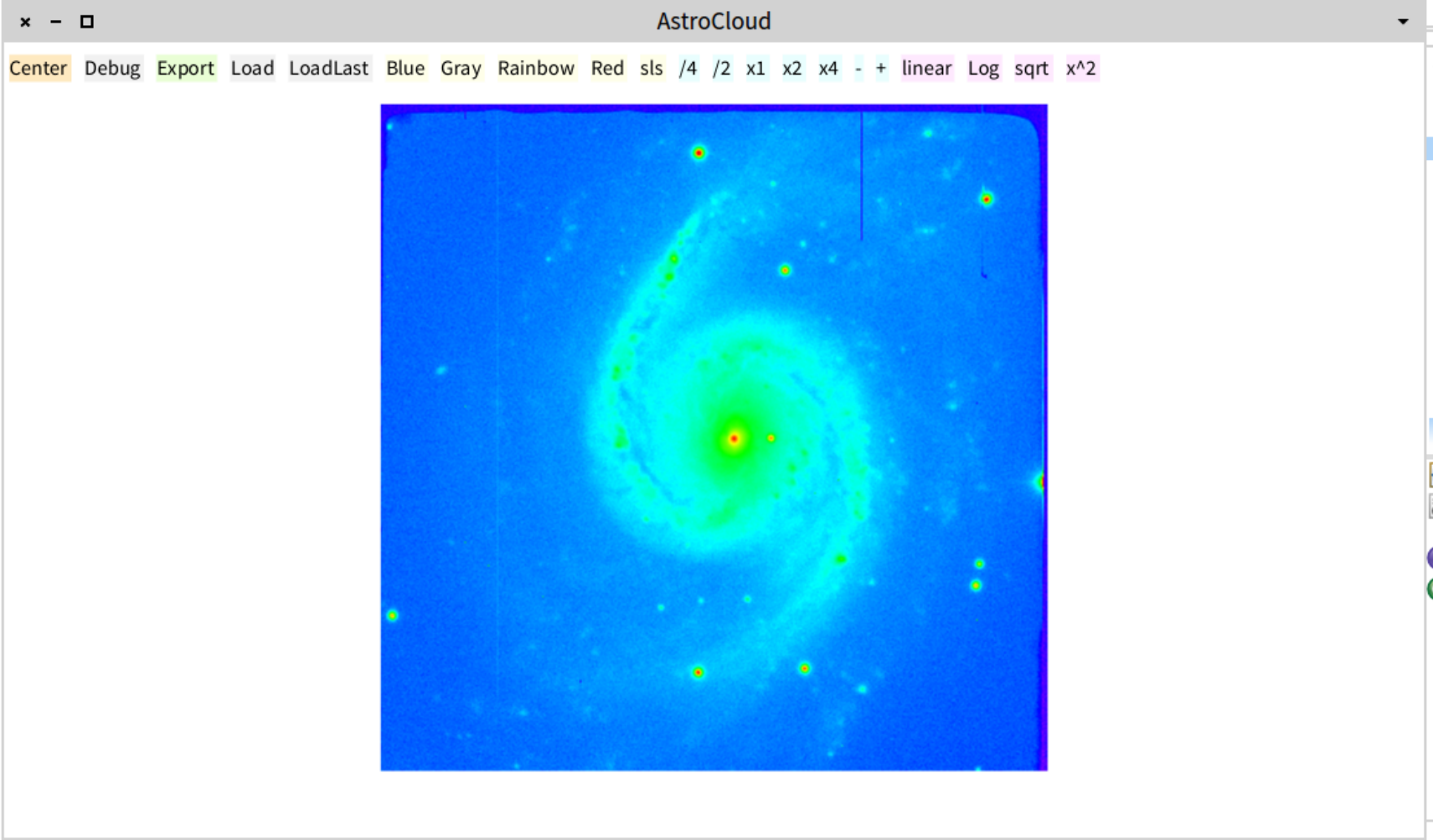
Meta



```
b3 := RTPieBuilder new.  
b3 view: c view.  
b3 interaction popup.  
b3 objects: classesToAnalyze.  
es := b3 slice: #numberOfLinesOfCode.  
b3 globalRule  
  elements: es;  
  distinctColor.  
b3 build.  
c propagateHighlight.  
c group: #pie.  
  
"-----"  
b4 := RTCharterBuilder new.  
b4 view: c view.  
b4 extent: 290 @ 200.  
b4 interaction popup.  
b4 shape ellipse size: 15; color: (Color red alpha: 0.3).  
b4 points: classesToAnalyze.  
b4 x: #numberOfMethods; y: #numberOfVariables min: 0 max: 10.  
b4 size: #numberOfLinesOfCode min: 4 max: 20 using: #sqrt.  
  
b4 axisXWithNumberOfTicks: 3.  
b4 axisYWithNumberOfTicks: 4.  
b4 highlightIf: [ :cls | (cls numberOfLinesOfCode > 2000) or:  
[ cls numberOfVariables > 10 ] ] using: #name.  
c propagateHighlight.  
c group: #plot.  
  
"-----"  
c move: #hierarchy onTheLeftOf: #pie.  
c move: #stat below: #hierarchy.  
c move: #plot below: #stat.  
  
c nameGroup: #hierarchy as: 'class hierarchy'.  
c nameGroup: #stat as: '#methods'.  
"c nameGroup: #pie as: '#LOC'."
```

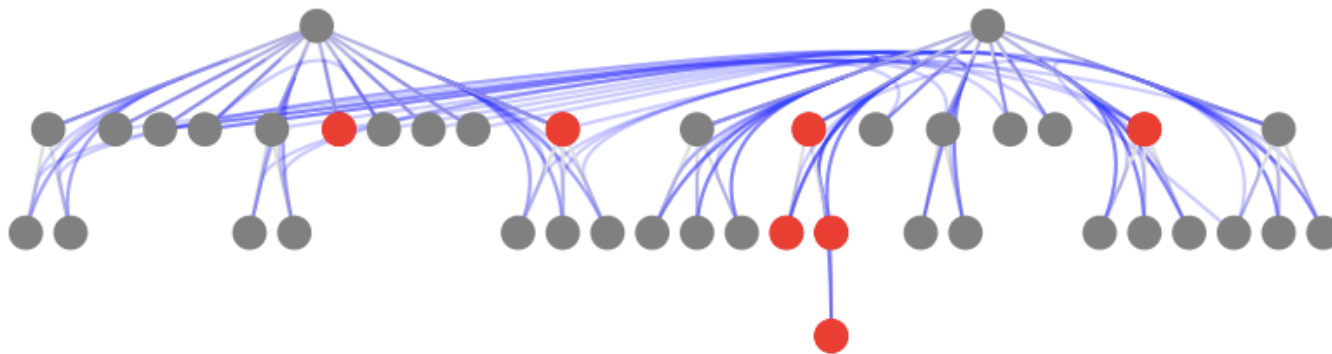
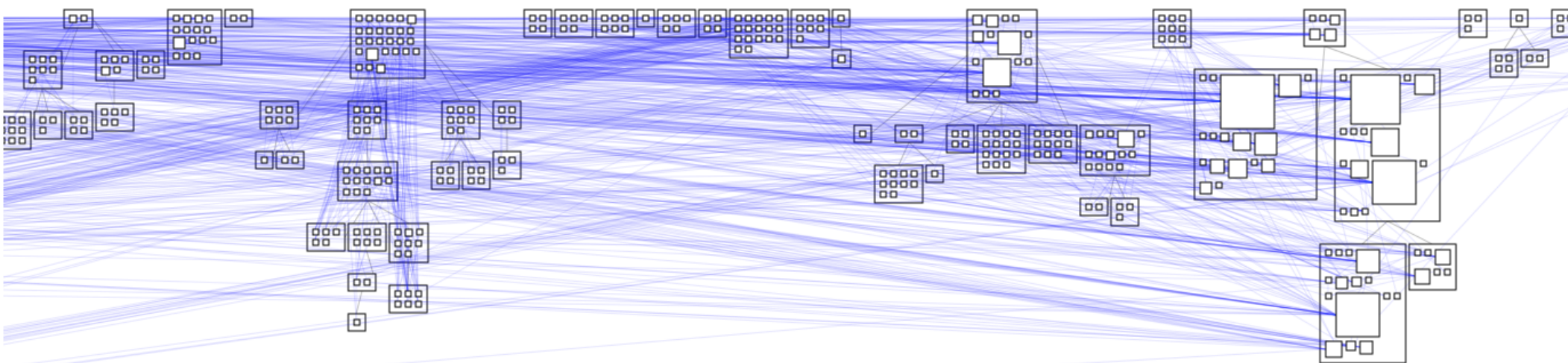


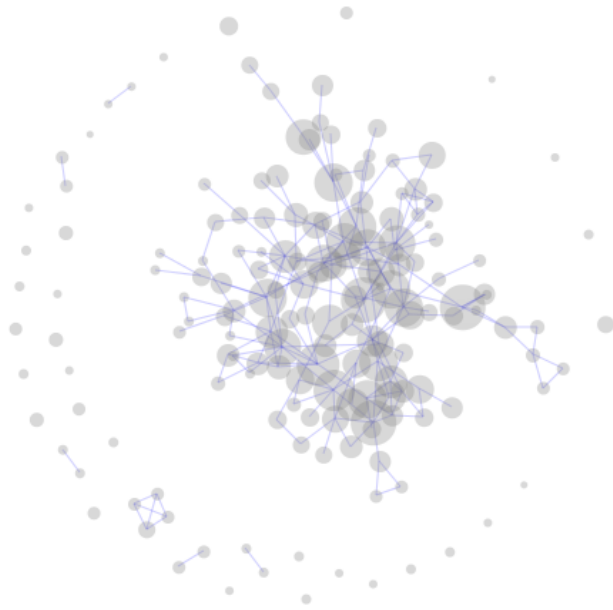
moosetechnology.org



Roassal visualization

Find... Zoom in Zoom out Export Run tests Refresh





<http://objectprofile.com>

<https://www.facebook.com/ObjectProfile>

