

Nour Jihene Agouf: Ph.D. Student

Company & Team Lab: Arolla &
RMod/Evref




Contact email: nour-jihene.agouf@inria.fr

Website: www.nouragouf.fr

Nour Jihene Agouf. Publications Research Teaching Activities Blogs

About me

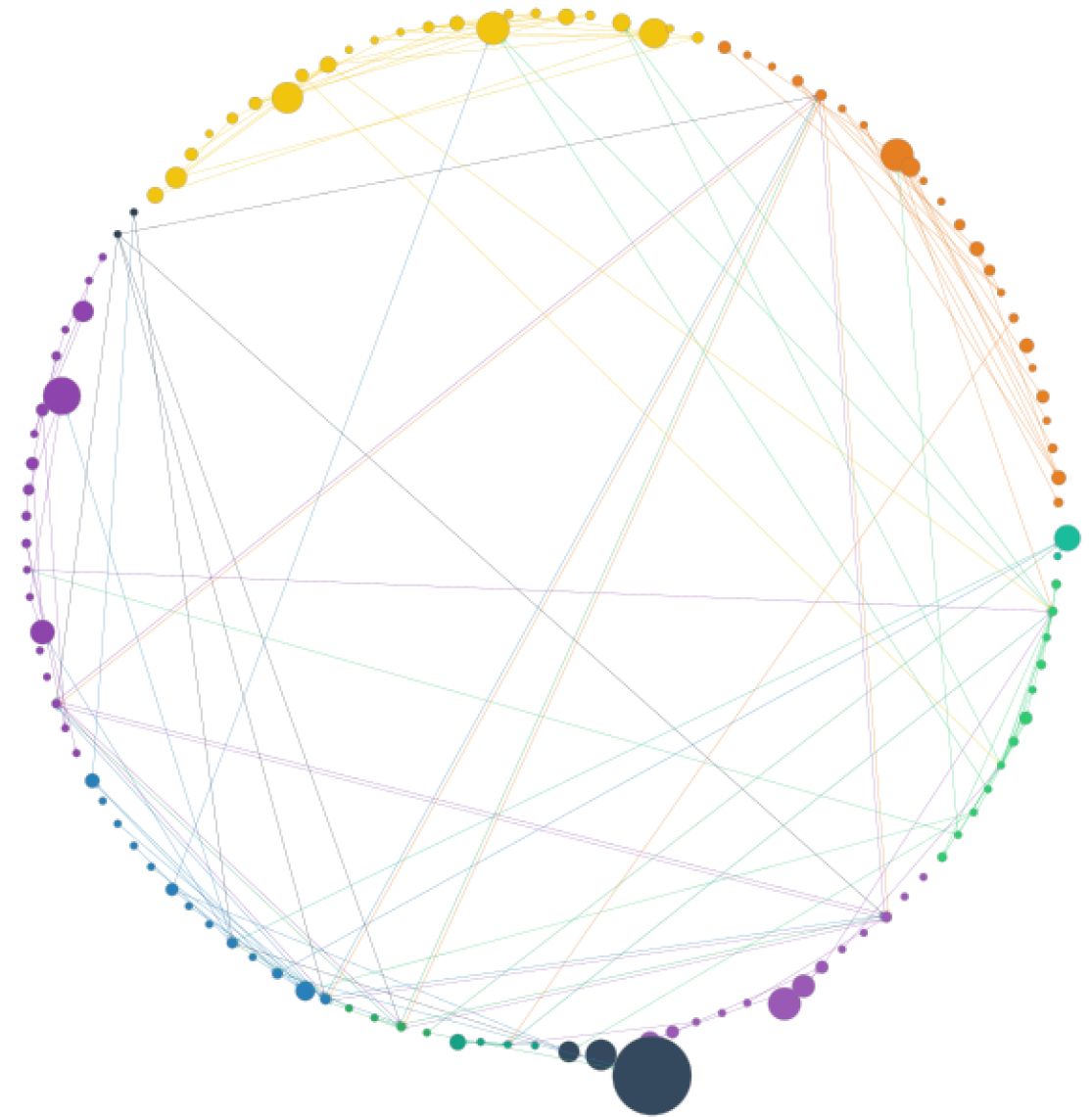
I am Research Engineering at Arolla practicing my Ph.D within the RMod team, which is joint to CNRS, Inria. Since I started my Ph.D. in 2021, my research included working on using and deploying visualizations for legacy software to comprehend and assess the quality of the code base for maintainable and durable software.



Roassal

- In the Pharo environment
- Data visualization
- A set of tools

```
"Animations"  
RSTAnimationExamples new example24RoassalPerlinNoise open extent: 800@500.  
RSTAnimationExamples new example09PerlinParticles open extent: 800@500.  
"Charts"  
RSChartExample new example24SpineLine open.  
RSKiviatExample new example06Chemistry open.  
"Images"  
RSSVGAnimationExamples new example02Miku open.  
"System complexity"  
MySystemComplexity new example open extent: 900@600
```



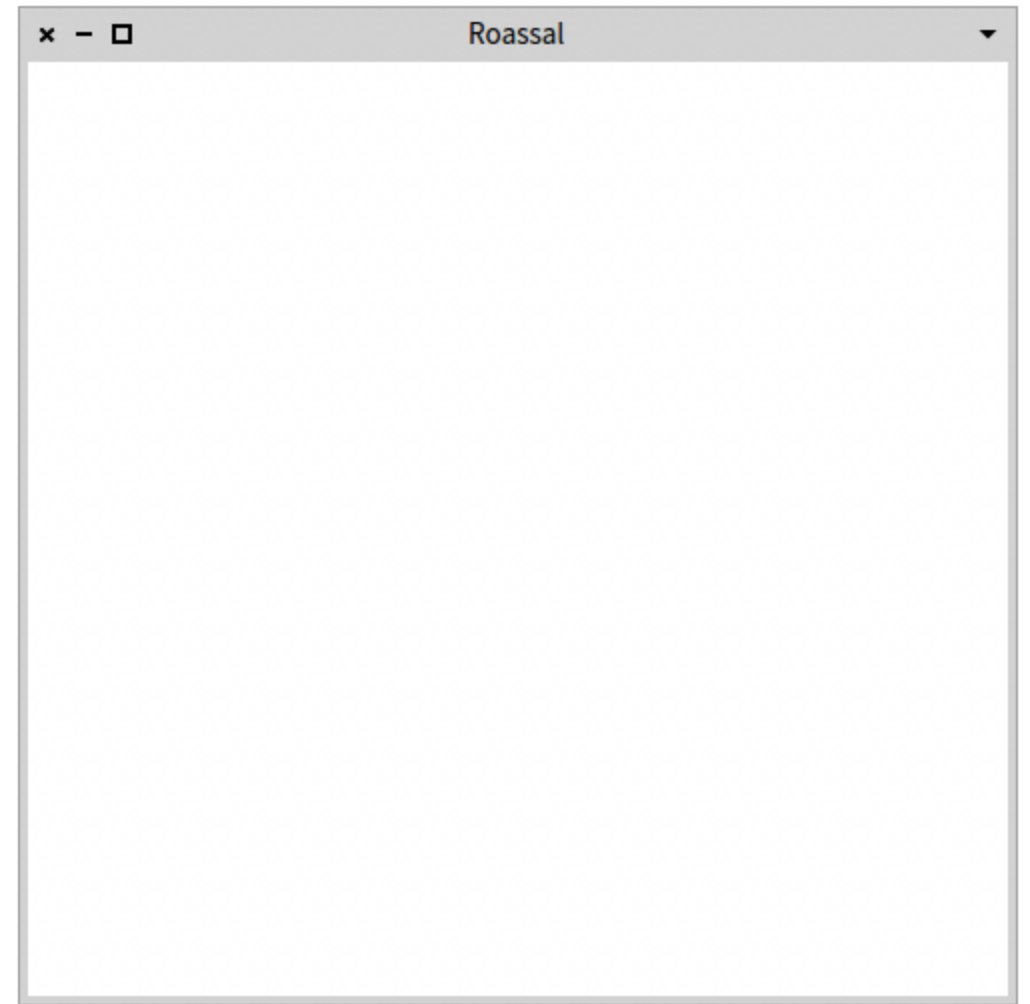
Roassal - Main components

- Canvas
- Shapes
- Layouts
- Color Palettes
- Normalizer
- Links
- Interactions
- Events

The canvas

- Contains and displays shapes
- The origine (0@0) is located at the center

```
canvas := RSCanvas new.  
canvas open
```



Shapes

- Subclasses of `RSShape`
 - Rectangle : `RSBox`
 - Cercle : `RSCircle`
 - Ligne : `RSLine`
 - Texte : `RSLabel`
 - etc.

Shapes

- Rectangle

```
rect := RSBox new.
```

- Cercle

```
circle := RSCircle new.
```



Edit the shapes

Properties: #height: , #width: , #size: ,
#color: , #border: , #borderColor:

```
rect height: 50;  
width: 100;  
color: Color red;  
border: RSBorder new.
```



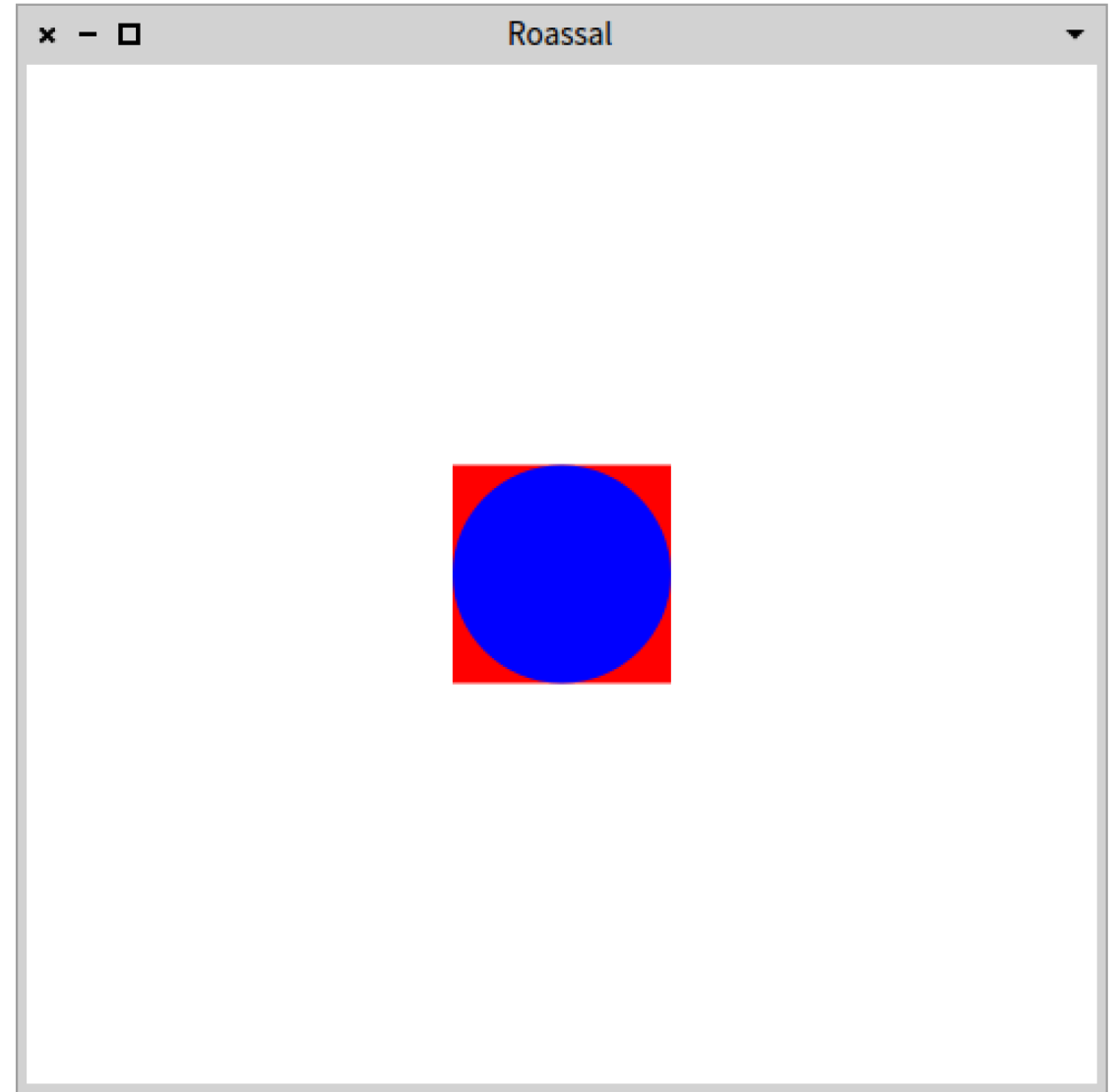
Shapes

- Associate a user data to a shape (model)
 - A shape can represent a Pharo object
 - Actions to a shape according to the represented object

```
shape model: 1
```


Shapes in the canvas

```
canvas add: rect.  
canvas add: circle.  
"canvas addAll: {rect . circle}"  
canvas open
```



Application

For every method of the package `Roassal3-Builders` classes, create a circle that describes it.

The result must be a set of shapes.

Add these shapes to a canvas and open the canvas.

Layouts

Allows to manage the disposition of these objects in the canvas.

The subclasses of the class `RSLayout` :

- Horizontal display `RSHorizontalLineLayout`
- Vertical display `RSVerticalLineLayout`
- Hierarchical display `RSTreeLayout`
- Circular display `RSCircleLayout`
- etc.

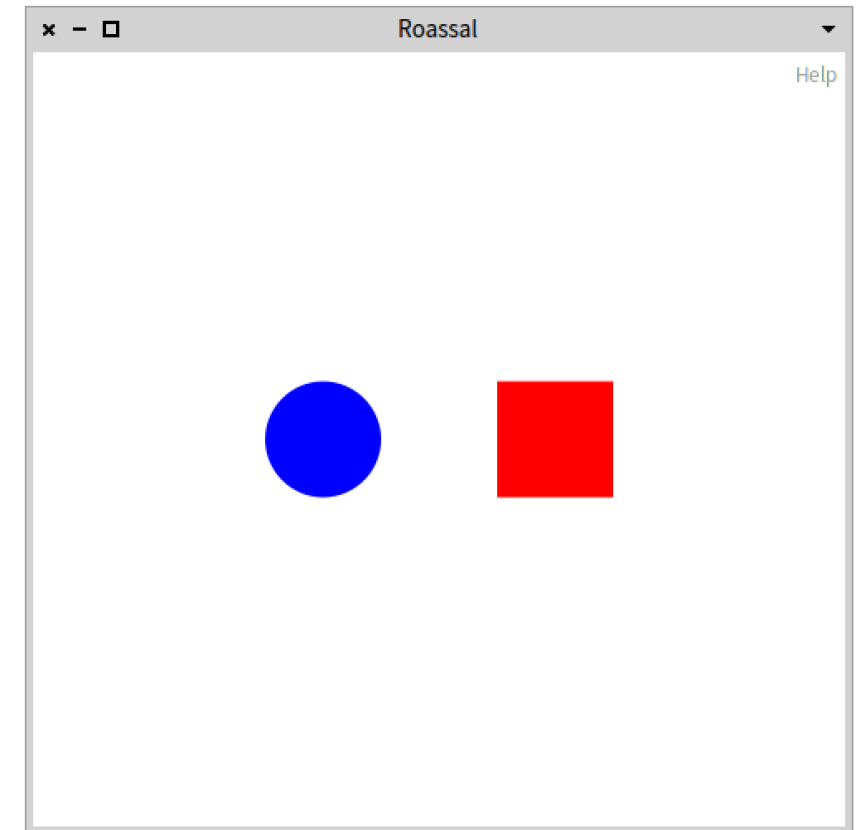
```
RSAnimationExamples new example33AnimatedLayout open
```

Layouts

- Horizontal display

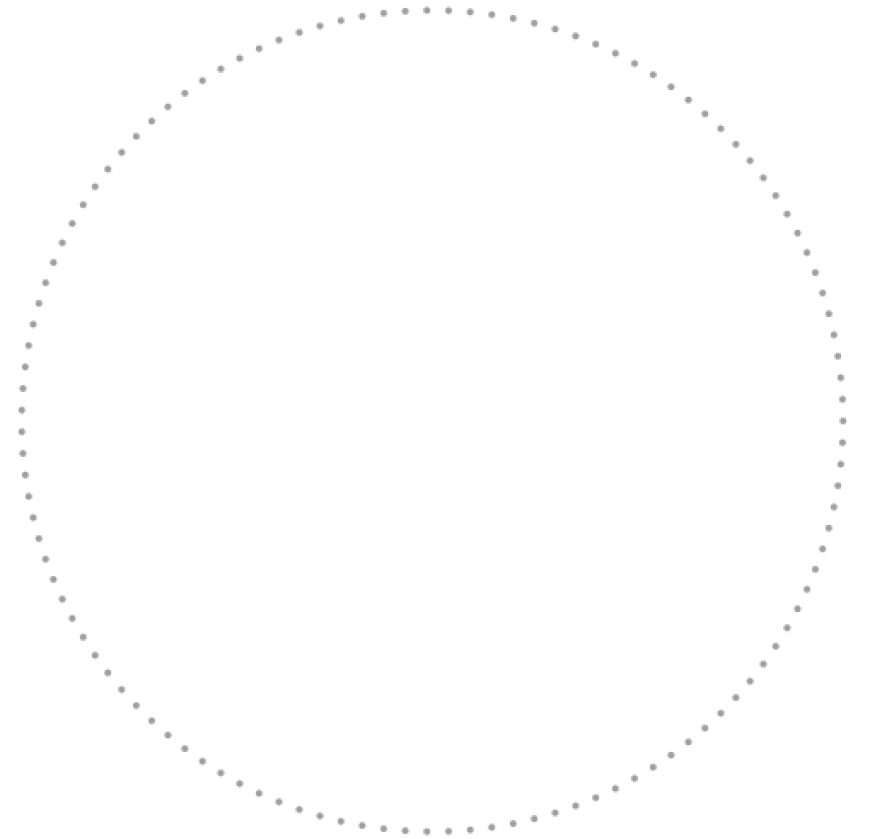
`RSHorizontalLineLayout`

```
RSHorizontalLineLayout on: {circle . rect}.  
canvas add: circle;  
    add: rect.  
canvas open
```



Application

- Add a layout to the canvas shapes.

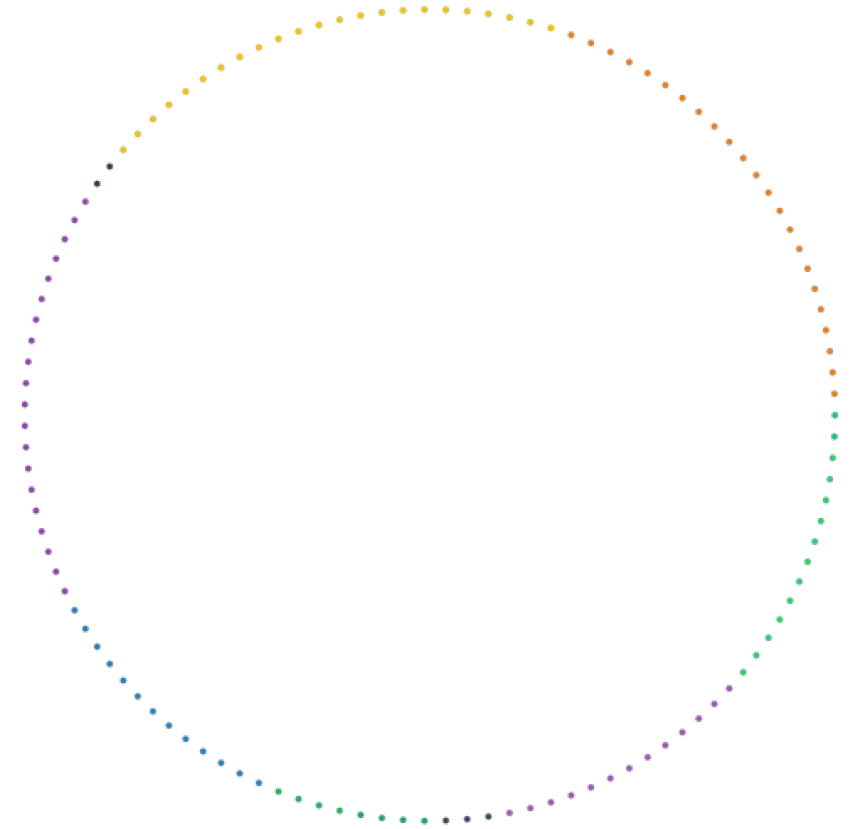


Color Palettes

- Subclasses of the `RSColorPalette` class
- Offers multiple palettes
 - `RSDivergingColorPalette`, `RSQualitativeColorPalette`,
`RSSequentialColorPalette`

Application

- Attribute to each class a color describing it.
- Color each methods shape according to its class (methods of the same class must have the same color)
 - To do so, select a color palette with the same number of classes.



Normalizer

- is a common task in various computer graphics and pattern recognition applications. It aims to normalize different objects into a canonical coordinate frame in order to guarantee a unique representation

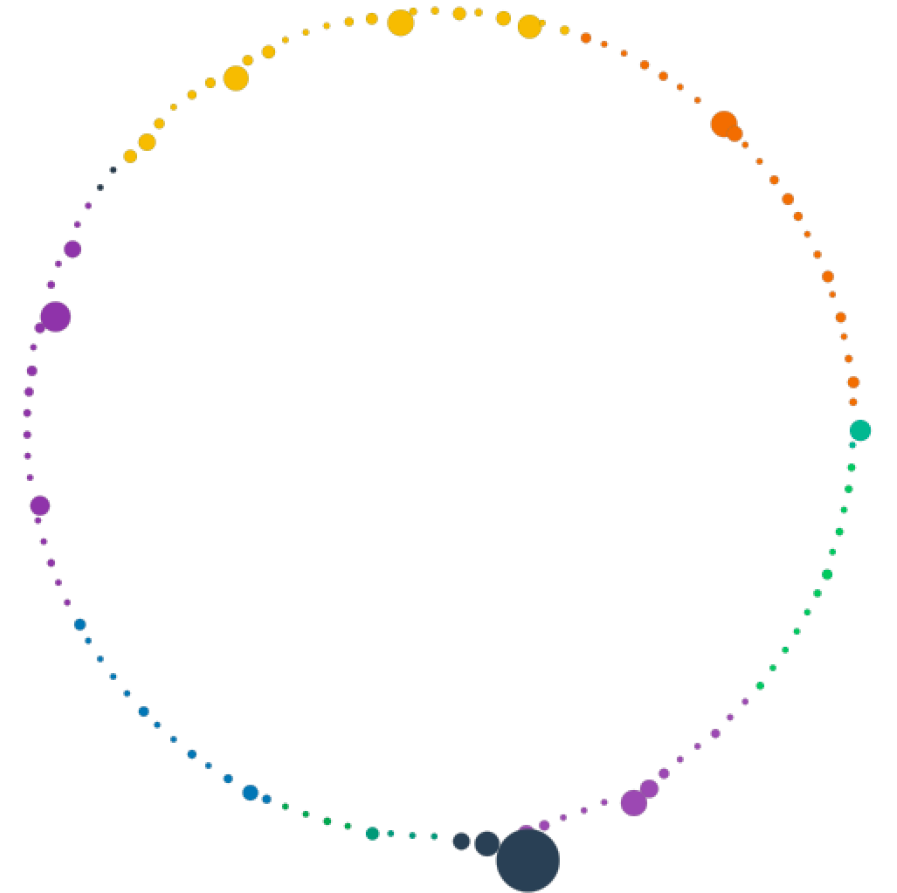
Normalizer

Example:

```
c := RSCanvas new.  
  
(30 to: 100 by: 5) do: [ :nb |  
    c add: (RSEllipse new size: nb; model: nb) ].  
  
RSNormalizer size  
    shapes: c shapes;  
    normalize: #yourself.  
  
RSNormalizer color  
    shapes: c shapes;  
    normalize: #yourself.  
RSFlowLayout on: c shapes.  
c shapes @ RSPopup.  
c @ RSCanvasController
```

Application

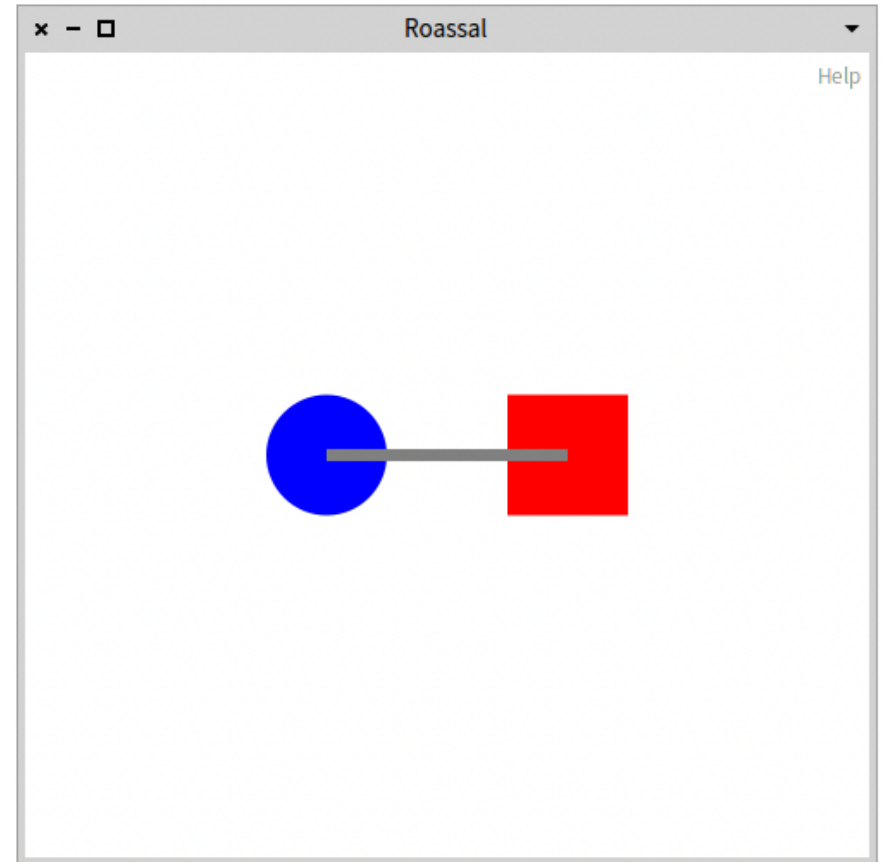
- Normalize the shape of each method according to its number of lines of code.



Links

- Links shapes

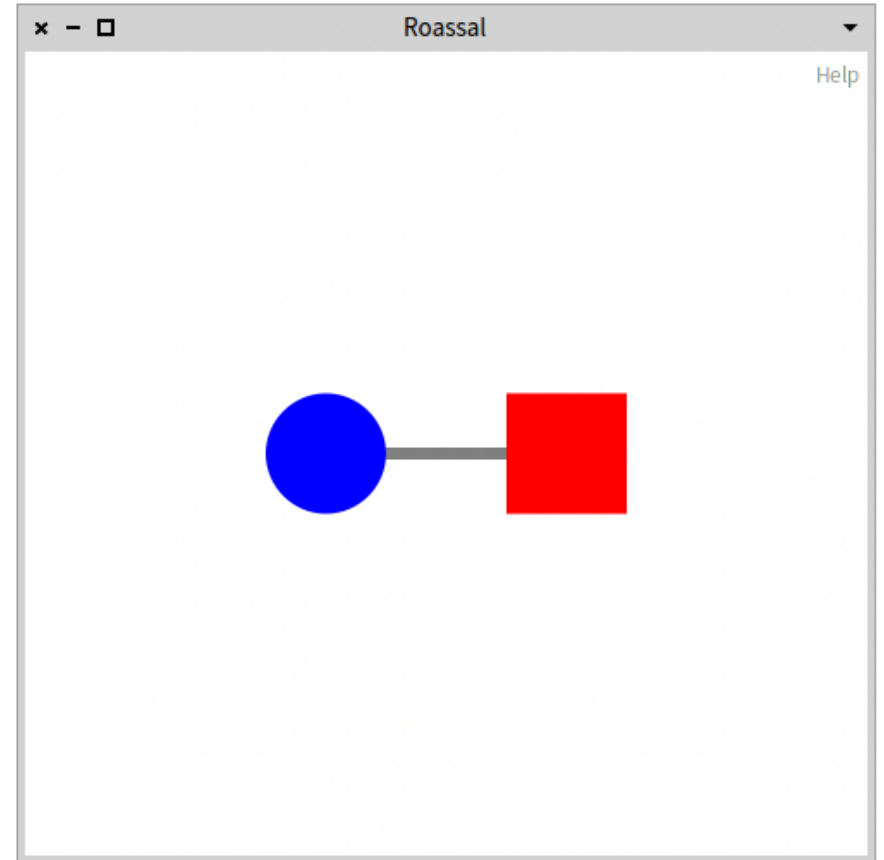
```
line := RSLine new.  
line from: rect;  
      to: circle.  
canvas add: line.
```



Links

- With a different attachment point

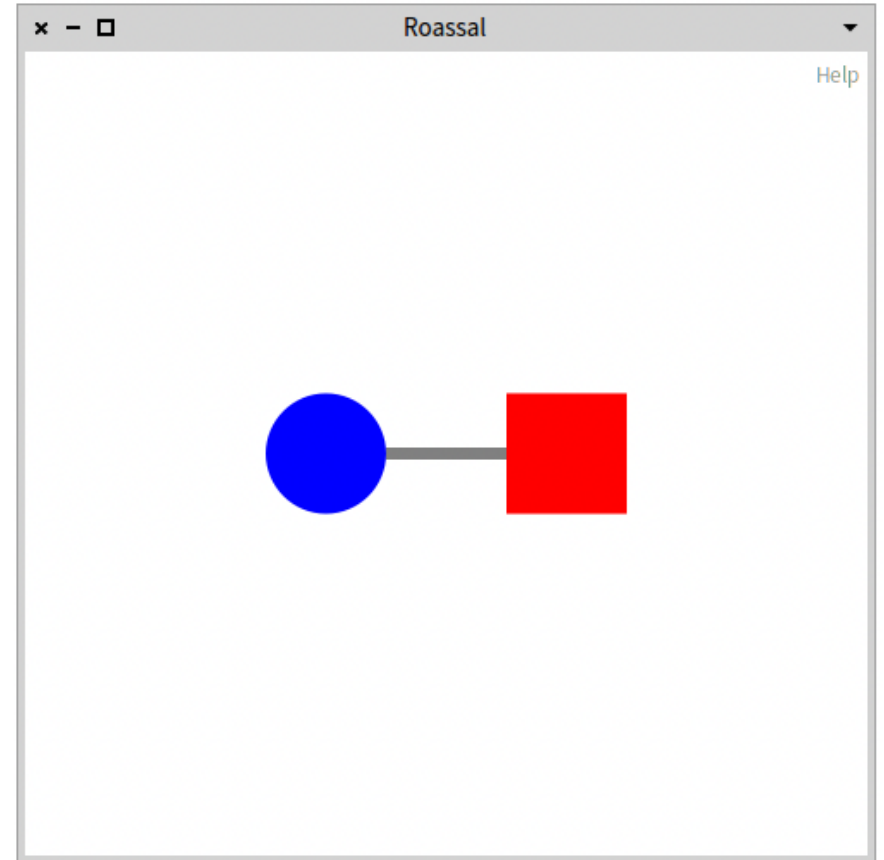
```
line := RSLine new.  
line withBorderAttachPoint;  
  from: rect;  
  to: circle.  
canvas add: line.
```



Links

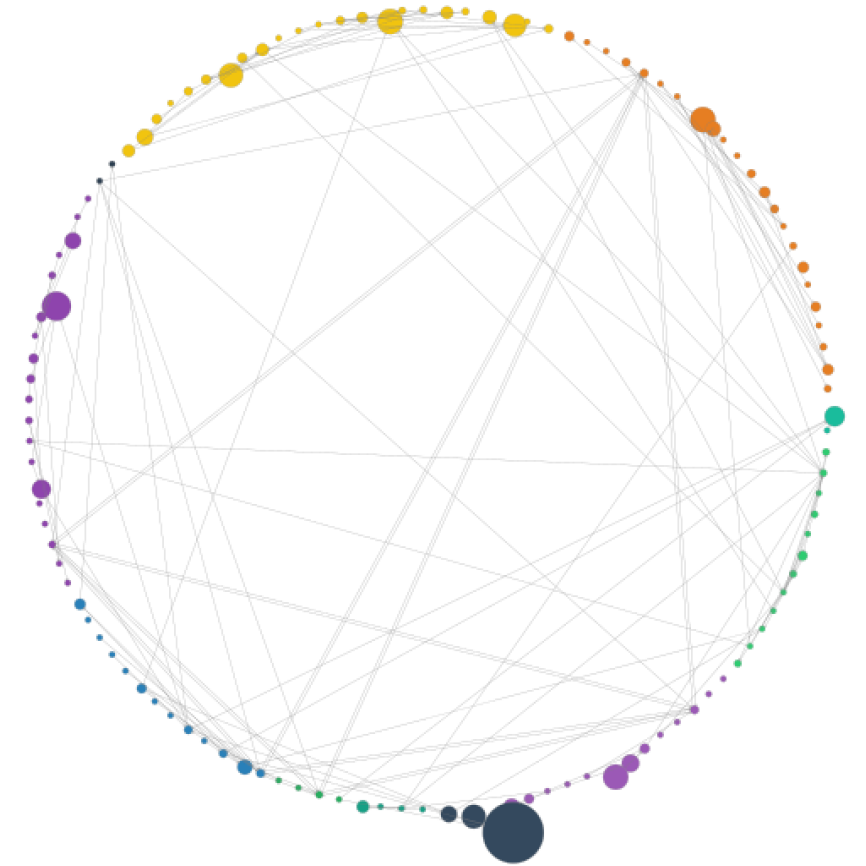
- With a builder

```
RSLineBuilder line  
  canvas: c;  
  connectFrom: [ :model | ].
```



Application

- Connect methods that call each other



Interactions

Subclasses of the class `RSInteraction` :

- Draggable `RSDraggable`
- Popup `RSPopup`
- Highlight `RSHighlightable`
- Menu `RSMenuActivable`
- etc.

```
shape @ RSPopup "Display the name of the model when a mouse hover"
```

Application

Add an interaction to the methods shapes:

- Make all shapes draggable.
- Create a popup when a mouse hover the shape that displays the name of the method class, the method name, number of lines of code, and the number of senders.

Events

- Subclasses of the class `RSEvent` .
 - `RSMouseClick` , `RSMouseEnter` , `RSKeyDown` , etc.

```
shape on: RSEvent do: [ :evt | "Action à réaliser" ]
```

Application

- Add events to each shape, allowing to inspect the method when a mouse click.
- Color the method and its senders in red when a mouse hover.
- Revert when mouse leaves the shape.

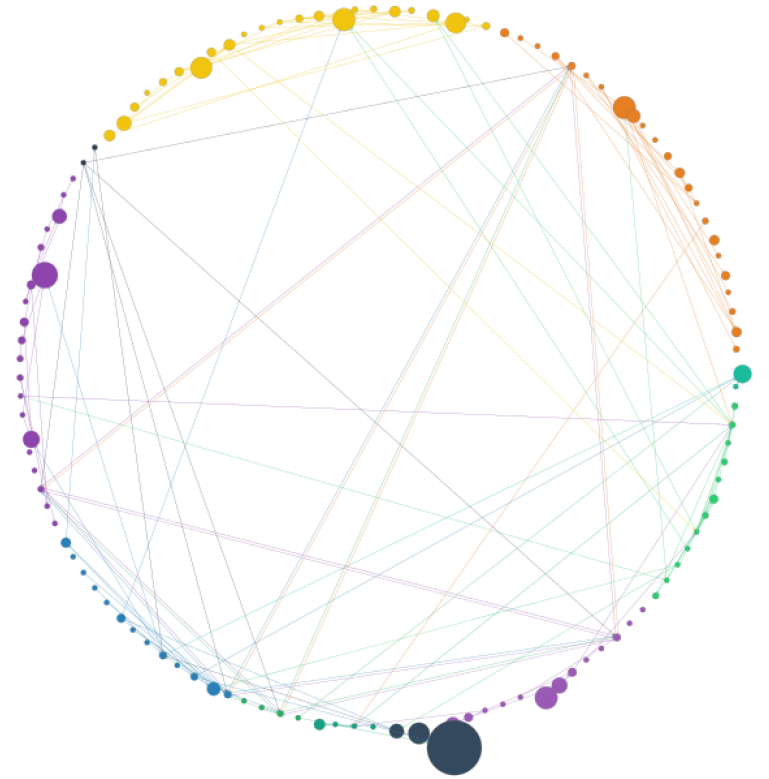
Some Roassal tools

- Exporters (pdf, svg, png, ...)

```
RSPNGExporter new  
    canvas: self;  
    filename: 'myCanvas';  
    export
```

Application

- Color edges the same as their sources.
- Add a green border for abstract methods.



Ressources

- Github (MIT)
 - <https://github.com/ObjectProfile/Roassal3>
- Documentation
 - <https://github.com/ObjectProfile/Roassal3Documentation>
- Exporters
 - <https://github.com/ObjectProfile/Roassal3Exporters>
- Agile Visualization
 - <http://agilevisualization.com/>