Chapter 1

Coloring

Colors plays an important role in a visualization. It may indicate particular situations without requiring a long visual processing time. This mechanism is called *pre-attentive perception* and is key to efficiently and quickly conveying information.

Roassal offers several mechanisms to easily associate colors to elements. This chapter gives an overview of coloring in Roassal.

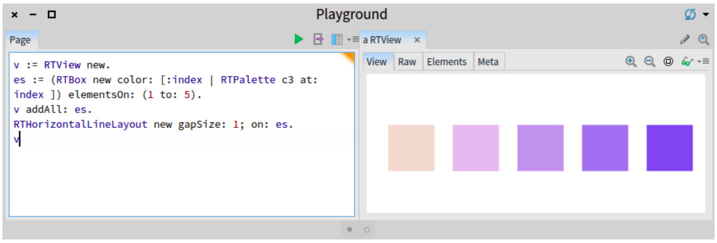
1.1 Palettes

Roassal contains a number of color schemas, available in the class RTPalette. Consider the example (Figure 1.1):

```
v := RTView new.
es := (RTBox new color: [ :index | RTPalette c3 at: index ]) elementsOn: (1 to: 5).
v addAll: es.
RTHorizontalLineLayout new gapSize: 1; on: es.
v
```

This simple example creates a view referenced by the variable v. A shape is created by instantiating the class RTBox and setting a block as a color. This block is evaluated for each element. Elements are created with elementsOn: (1 to: 5): five elements are created, each having a number as the element model. The shape evaluates the block provided as a color by using the model as the index variable. The palette given by RTPalette c3 has 5 different colors, represented as an Array of the same size.

The class RTPalette has several palettes. You can replace the c3 from any value ranging from c1 to c14.



```
v := RTView new.
es := (RTBox new color: [:index | RTPalette c3 at:
index ]) elementsOn: (1 to: 5).
v addAll: es.
RTHorizontalLineLayout new gapSize: 1; on: es.
v
```

Figure 1.1: Example of a palette.

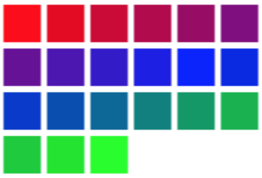


Figure 1.2: Simple coloring.

1.2 Multi-linear color normalizer

In the Roassal jargon, a normalizer is a facility to map a set of objects over a range of values or colors. Several color normalizers are provided to color an element according to some particular requirements.

The RTMultiLinearColor class is a simple and useful color normalizer. It associates a color to each element depending on a numerical value, ranging from 0.0 to 1.0. Consider the following example (Figure 1.2):

```
v := RTView new.
n := RTMultiLinearColor new.
shape := RTBox new size: 30; color: n.
elements := shape elementsOn: (0.0 to: 1.0 by: 0.05).

v addAll: elements.
RTGridLayout on: elements.
v
```

The code given above first creates a view. It then creates an instance of RTMultiLinearColor. This normalizer is polymorph to the class Color, which means you can provide a color normalizer where a color or a block is usually expected. The example simply uses the normalizer as a color by being provided to color:.

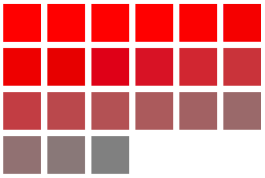


Figure 1.3: Simple coloring.

The default configuration of RTMultiLinearColor is to map a value ranging from 0.0 to 1.0 to the default color palette, composed of red, blue, and green. The elements created with shape elementsOn: (0.0 to: 1.0 by: 0.05) have a numerical value as model. This value is directly mapped to a color.

A particular palette may be set using color:. The following example maps the value from red to gray (Figure 1.3):

```
v := RTView new.
n := RTMultiLinearColor new.
n colors: { Color red . Color gray }.
shape := RTBox new size: 30; color: n.
elements := shape elementsOn: (0.0 to: 1.0 by: 0.05).

v addAll: elements.
RTGridLayout on: elements.
v
```

The example above uses a palette made of two colors, red and gray. These two colors are then interpolated to give a value ranging from 0.0 and 1.0. The value 0.0 is mapped to red, and 1.0 to gray.

In the examples given above each element has a numerical value as object model. It does not have to be so. Consider the following example (Figure 1.4):

```
classes := Collection withAllSubclasses.
maximumNumberOfLinesOfCode := (classes collect: #numberOfLinesOfCode)
   max.

v := RTView new.
n := RTMultiLinearColor new.
n colors: RTPalette c4.
n command: [ :cls | cls numberOfLinesOfCode / maximumNumberOfLinesOfCode
    ].

shape := RTBox new color: n.
elements := shape elementsOn: classes.
```

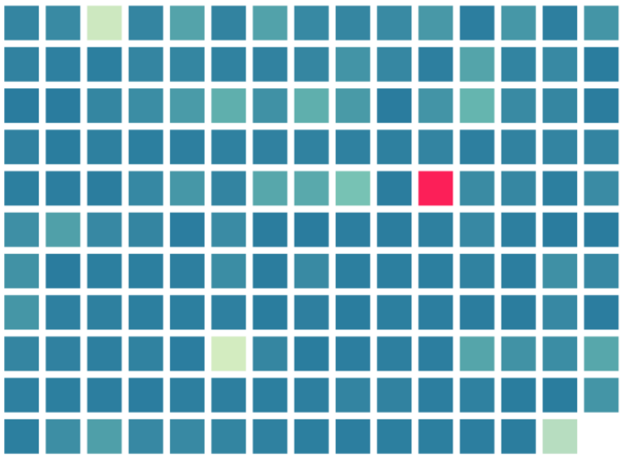


Figure 1.4: Classes and their size indicated using a color.

```
v addAll: elements.
RTGridLayout new gapSize: 1; on: elements.

v
```

The variable classes contains a collection of classes to be represented. The largest number of lines of code is obtained from the collection classes. The message max returns the maximum value of a set of numerical values (*e.g.,* #(1 2 42 3 4) max returns 42).

The normalizer n uses a palette as the colors to be used. The message command: takes a block or a symbol as argument to compute the color index. The block value has to return a numerical value ranging from 0.0 to 1.0.

The visualization can be enriched with edges and a dedicated legend (Figure 1.5):

```
classes := Collection withAllSubclasses.
maximumNumberOfLinesOfCode := (classes collect: #numberOfLinesOfCode)
   max.

v := RTView new.
n := RTMultiLinearColor new.
n colors: RTPalette c4.
n command: [ :cls | cls numberOfLinesOfCode / maximumNumberOfLinesOfCode
    ].

shape := RTEllipse new size: 13; color: n.
elements := shape elementsOn: classes.
```

```
v addAll: elements.

eb := RTEdgeBuilder new.
eb view: v.
eb moveBehind.
eb connectFrom: #superclass.

RTClusterLayout new on: elements.

"Add the legend"
lb := RTLegendBuilder new.
lb view: v.
lb addText: 'Each circle is a class'.
lb addText: 'Edges indicate inheritance between classes'.
lb addText: '(Subclasses are closer to the border than their superclass)'.
lb addColorFadingUsing: n colors text: 'Number of lines of code'.
lb build.

v
```

It is common to have to normalize colors according to a particular metric. In the previous example, we did it manually as we needed to compute the maximum number of lines of code. Computing maximum and minimum values is often cumbersome.

Roassal offers a dedicated infrastructure, called RTMetricNormalizer, to ease the color normalization using a metric. Use of this normalizer makes unnecessary to compute the largest size in the set. Consider this new version (without legend to keep the code short):

```
classes := Collection withAllSubclasses.

v := RTView new.

shape := RTEllipse new size: 13.
elements := shape elementsOn: classes.

v addAll: elements.
RTMetricNormalizer new
   elements: elements;
   normalizeColor: #numberOfLinesOfCode using: RTPalette c4.

eb := RTEdgeBuilder new.
eb view: v.
eb moveBehind.
eb connectFrom: #superclass.

RTClusterLayout new on: elements.
```

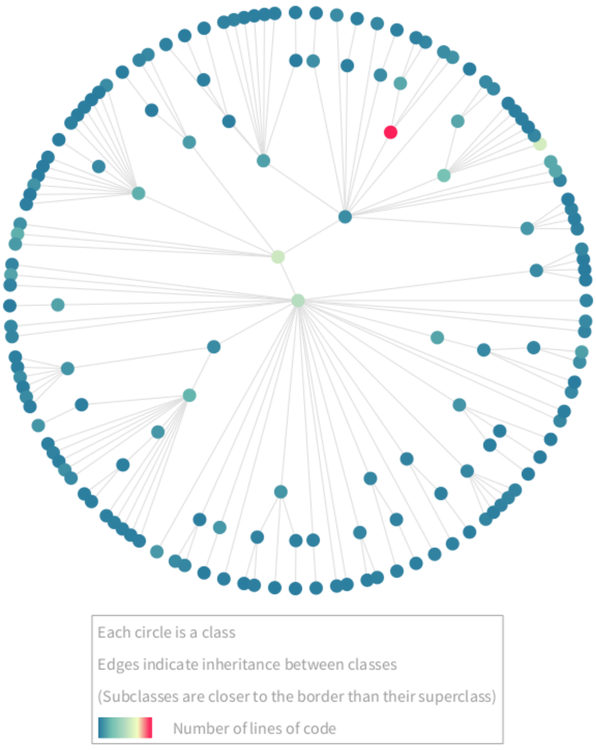

Each circle is a class
Edges indicate inheritance between classes
(Subclasses are closer to the border than their superclass)
Number of lines of code

Figure 1.5: Adding a legend.

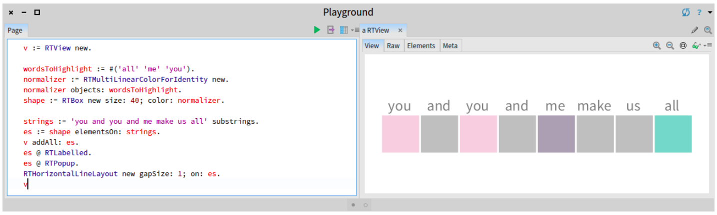
```
v
```

1.3 Coloring objects

The class RTMultiLinearColorForIdentity colors a Roassal element based on the object model. A color is given to each object model. Elements representing the same object will have the same color. Consider the following example (Figure 1.6):

```
v := RTView new.

wordsToHighlight := #('all' 'me' 'you').
normalizer := RTMultiLinearColorForIdentity new.
normalizer objects: wordsToHighlight.
shape := RTBox new size: 40; color: normalizer.
```



```
v := RTView new.

wordsToHighlight := #('all' 'me' 'you').
normalizer := RTMultiLinearColorForIdentity new.
normalizer objects: wordsToHighlight.
shape := RTBox new size: 40; color: normalizer.

strings := 'you and you and me make us all' substrings.
es := shape elementsOn: strings.
v addAll: es.
es @ RTLabelled.
es @ RTPopup.
RTHorizontalLineLayout new gapSize: 1; on: es.
v
```

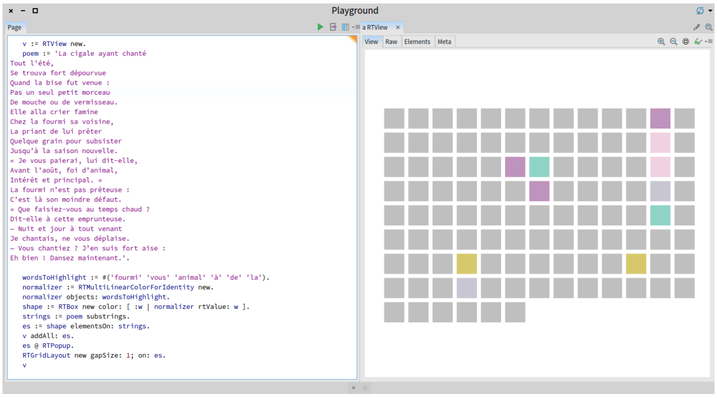
Figure 1.6: Example of RTMultiLinearColorForIdentity.

```
strings := 'you and you and me make us all' substrings.
es := shape elementsOn: strings.
v addAll: es.
es @ RTLabeled.
es @ RTPopup.
RTHorizontalLineLayout new gapSize: 1; on: es.
v
```

The example above represents a set of words and highlights some of them. The variable wordsToHighlight contains the words that have to be highlighted. The normalizer is configured with the words using objects:. The elements to display are created by having to cut the string 'you and you and me make us all' into single words using #substrings.

Consider the following example using the famous French poem by Jean de La Fontaine (Figure 1.7)

```
  v := RTView new.
  poem := 'La cigale ayant chante
Tout l''ete,
Se trouva fort depourvue
Quand la bise fut venue :
Pas un seul petit morceau
De mouche ou de vermisseau.
Elle alla crier famine
Chez la fourmi sa voisine,
La priant de lui preter
Quelque grain pour subsister
Jusqu''a la saison nouvelle.
" Je vous paierai, lui dit-elle,
Avant l''aout, foi d''animal,
Interet et principal. "
La fourmi n''est pas preteuse :
C''est la son moindre defaut.
" Que faisiez-vous au temps chaud ?
Dit-elle a cette emprunteuse.
```



```
   v := RTView new.
   poem := 'La cigale ayant chanté
Tout l’été,
Se trouva fort dépourvue
Quand la bise fut venue :
Pas un seul petit morceau
De mouche ou de vermisseau.
Elle alla crier famine
Chez la fourmi sa voisine,
La priant de lui prêter
Quelque grain pour subsister
Jusqu’à la saison nouvelle.
« Je vous paierai, lui dit-elle,
Avant l’août, foi d’animal,
Intérêt et principal. »
La fourmi n’est pas prêteuse :
C’est là son moindre défaut.
« Que faisiez-vous au temps chaud ?
Dit-elle à cette emprunteuse.
– Nuit et jour à tout venant
Je chantais, ne vous déplaise.
– Vous chantiez ? J’en suis fort aise :
Eh bien ! Dansez maintenant.'.

   wordsToHighlight := #('fourmi' 'vous' 'animal' 'à' 'de' 'la').
   normalizer := RTMultiLinearColorForIdentity new.
   normalizer objects: wordsToHighlight.
   shape := RTBox new color: [ :w | normalizer rtValue: w ].
   strings := poem substrings.
   es := shape elementsOn: strings.
   v addAll: es.
   es @ RTPopup.
   RTGridLayout new gapSize: 1; on: es.
   v
```

Figure 1.7: Visualizing a poem.

```
– Nuit et jour a tout venant
Je chantais, ne vous deplaise.
– Vous chantiez ? J''en suis fort aise :
Eh bien ! Dansez maintenant.'.

   wordsToHighlight := #('fourmi' 'vous' 'animal' 'a' 'de' 'la').
   normalizer := RTMultiLinearColorForIdentity new.
   normalizer objects: wordsToHighlight.
   shape := RTBox new color: [ :w | normalizer rtValue: w ].
   strings := poem substrings.
   es := shape elementsOn: strings.
   v addAll: es.
   es @ RTPopup.
   RTGridLayout new gapSize: 1; on: es.
   v
```