**Learning about Roassal**

Roassal structures visualization in terms of views, elements, shapes, interactions and animations.

View is a container of elements. Interaction takes place between a user and elements viz. mouse hover, mouse click or keyboard interaction. Animations may include actions like popup of element, rotation or movement of elements.

Elements are graphical representation of a user defined objects. Elements can be given different shapes.

Model Object: These represents elements which have transparent update mechanism. I.e. As soon as object is modified; the visual representations of element are updated. Few examples are RTBox, RTElipse RTPolygon RTLabel etc.

An element can have few parameters viz. size, color, border color. Edges are primary part of Roassal. They connect different elements.

Elements can be grouped together so as they can be modified uniformly. This is done by using RTGroup.

A parent child hierarchy can be created between elements. This is done by nesting using the RTNesting functionality. The message “ On: elements1 nest: elements2” nests elements 2 under elements1.

User can interact with both view and elements. Interactions on view include RTDraggable View, RTHorizantalDraggable view, RTVerticalDraggable View and RTScrollBar which as name indicates can be used to drag view and scroll a view. There are many interaction wrt elements, to name a few RTPopup, RTDraggable, RTDraggableChildren.

Below I have listed few of the visualization that can be achieved using Roassal. We can have similar ones for our project.

**Double charting:**

|  |
| --- |
| tab := RTTabTable new input: (ZnEasy get: 'http://bit.ly/CensusGov') contents usingDelimiter: $,.  tab removeFirstRow.tab convertColumnsAsInteger: #('POPESTIMATE2013' 'POPEST18PLUS2013').  b := RTDoubleBarBuilder new.  b pointName: [ :row | row at: (tab indexOfName: 'NAME') ].  "Remove the first line, the sum"  b points: tab values allButFirst.  b bottomValue: [ :row | ((row at: (tab indexOfName: 'POPESTIMATE2013')) / 1000) asInteger ] titled: 'Pop estimate (x 1000)'.  b topValue: [ :row | ((row at: (tab indexOfName: 'POPEST18PLUS2013')) / 1000) asInteger] titled: 'Pop +18 estimate (x 1000)'.  b |

The above code takes input from the URL provided. Some operations are performed in order to fine tune the data so it is suitable to perform few operations and effectively visualize. For instance the first row is removed as the data is not necessary for current operation using removeFirstRow. RTDoubleBarBuilder provides a basic API which we can tailor to suit our visualization. The end result will be similar to the one provided in the image below.



**Graphing:**

|  |
| --- |
| tab := RTTabTable new input: (ZnEasy get: 'http://bit.ly/EbolaCSV') contents usingDelimiter: $,.  tab removeFirstRow.  tab replaceEmptyValuesWith: '0' inColumns: #(10 11).  tab convertColumnsAsInteger: #(10 11).  tab convertColumnsAsDateAndTime: #(3 4).  data := tab values reversed.  "Charting the data"  b := RTGrapher new.  ds := RTData new.  ds interaction fixedPopupText: [ :row | row value at: 12 ].  ds dotShape ellipse  color: (Color blue alpha: 0.3);  size: [ :row | (row at: 11) / 5 ].  ds points: data.  ds connectColor: Color blue.  ds y: [ :r | r at: 10 ].  ds highlightIf: [ :row | (row at: 10) > 100 ] using: [ :row | row third year ].  b add: ds.  b axisX noLabel; numberOfTicks: tab values size.  b axisY noDecimal.  b |

The above code defines a visualization which plots a graph with x,y axis.The input to this is provided through a excel file. The Roassal engine takes data from each cells and builds a visualization.Some operations such as convertColumnsAsDateAndTime are performed to suit the visualization. convertColumnsAsInteger provides values in integer format. The resulting visualization looks like below image.



**Integration with OpenStreetMap:**

OpenStreetMap is a collaborative project to create maps (http:*//*openstreetmap.org). One of the benefits of OpenStreetMap is to offers an API to access and download map tiles. We might as well use this when we use locations to plot visualization. Few of the visualization operations that can be integrated with this is shown below.

|  |
| --- |
| v := RTView new.  v @ RTDraggableView.  map := RTOSM new.  v add: map element.  "Place to set the data and center the camera"  london := 51.507222@ -0.1275.  Some arbitrary data"  data := ((1 to: 500) collect: [ :i | 50 atRandom - 25 ]) cumsum.  "We build the graph"  b := RTGrapher new.  b extent: 100@30.  d := RTData new.  d noDot.  d connectColor: Color red.  d points: data.  b add: d.  b axisY  labelFontHeight: 6;  color: Color red;  title: 'Sale'.  b axisX color: Color red; noTick; title: 'country'.  b build.  elementsAndEdges := b view elements, b view edges.  We create a white background" whiteBackground := (RTRoundedBox new color: Color white trans; borderRadius:  10) element.  v add: whiteBackground.  v addAll: elementsAndEdges.  RTNest new on: whiteBackground nest: elementsAndEdges.  whiteBackground translateTo: (map latLonToRoassal: london).  v canvas camera translateTo: (map latLonToRoassal: london).  v canvas camera noInitializationWhenOpen.  v canvas camera scale: 1.5.  v |

The above code takes location based on latitude and longitude. This plots the visualization based on data provided.



Updating the visualization should be done by manipulating trachel shapes. Roassal shapes create Roassal elements and a Roassal element creates a Trachel shape when added. Modifying a visualization is done by modifying trachel shapes or building appropriate Roassal interactions.

**References:**

All codes and images are taken from the book Agile Visualization. <http://agilevisualization.com/>

The codes and images are shared so as to share knowledge and familiarise teammates about new technology.