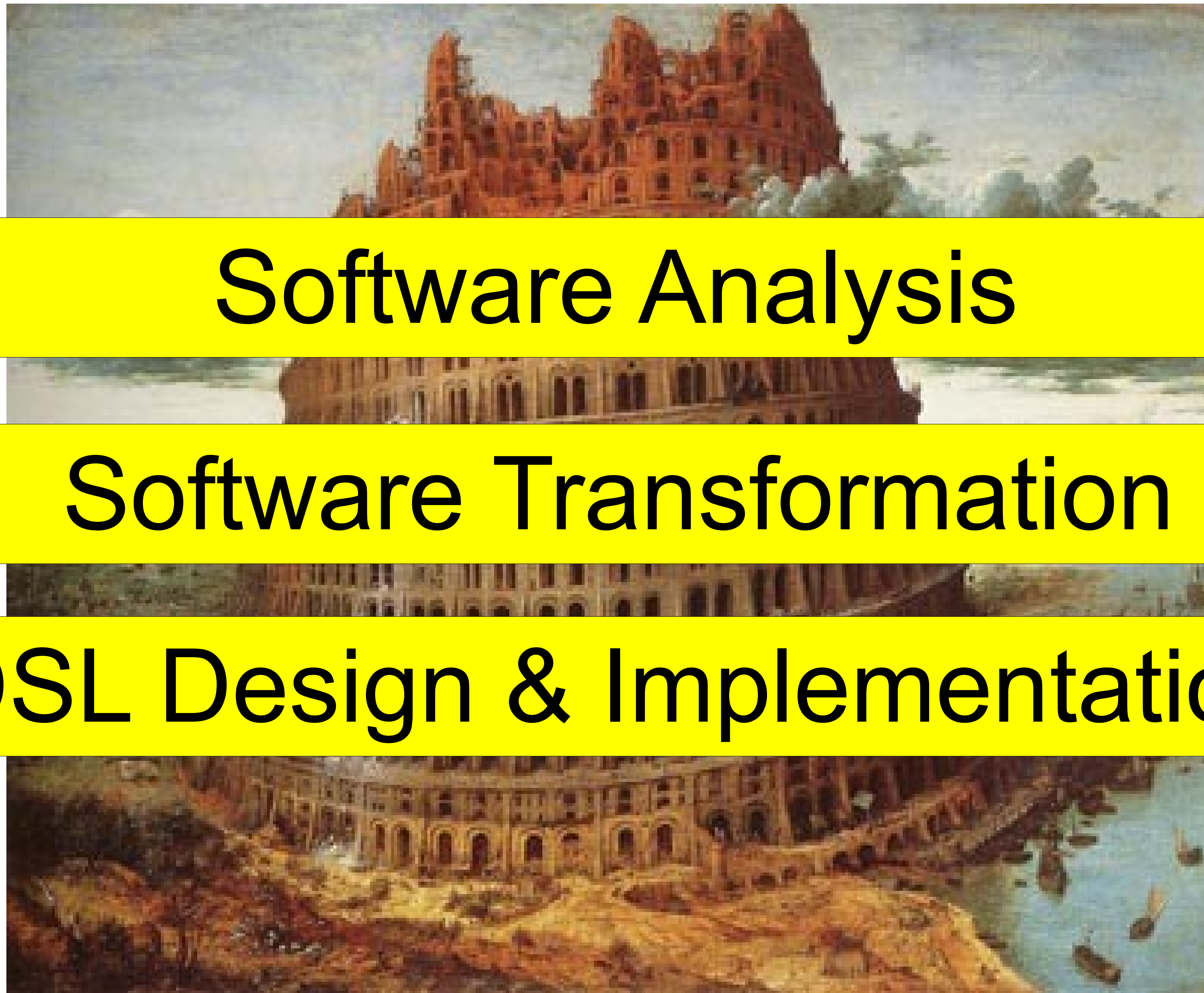
Towards Visual Software Analytics

Paul Klint

ATEAMS

Meta-Programming in Rascal



Software Analysis

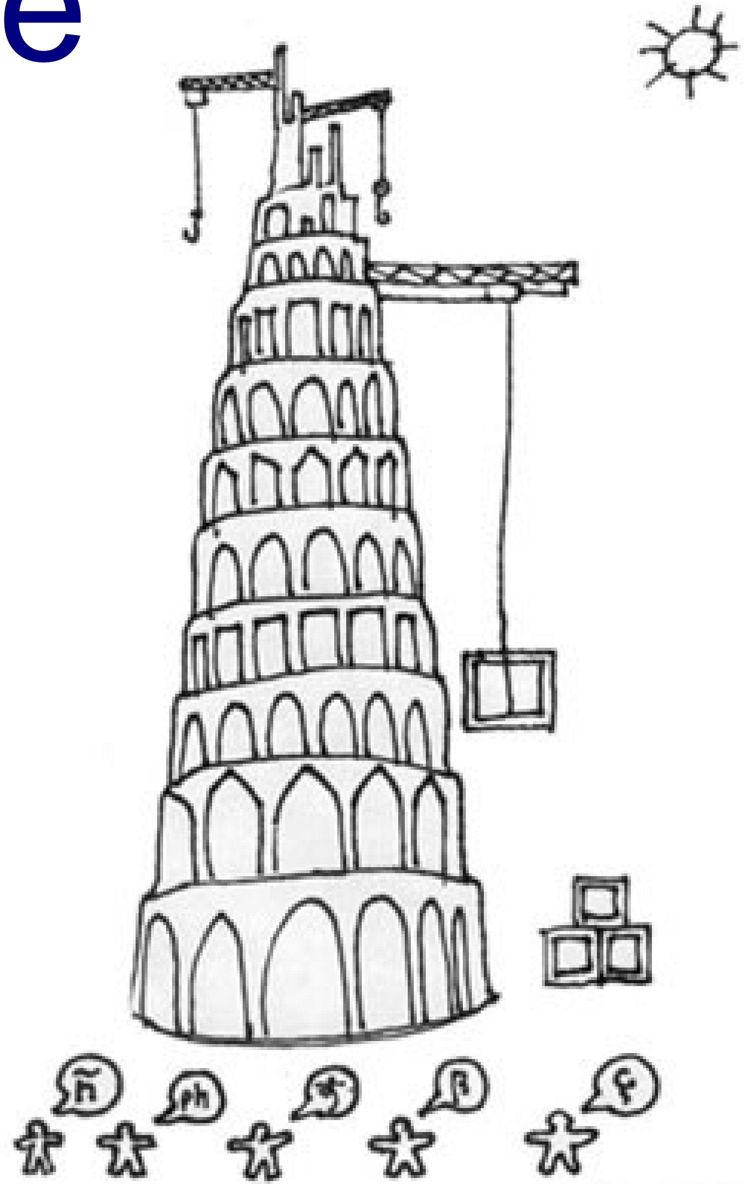
Software Transformation

DSL Design & Implementation

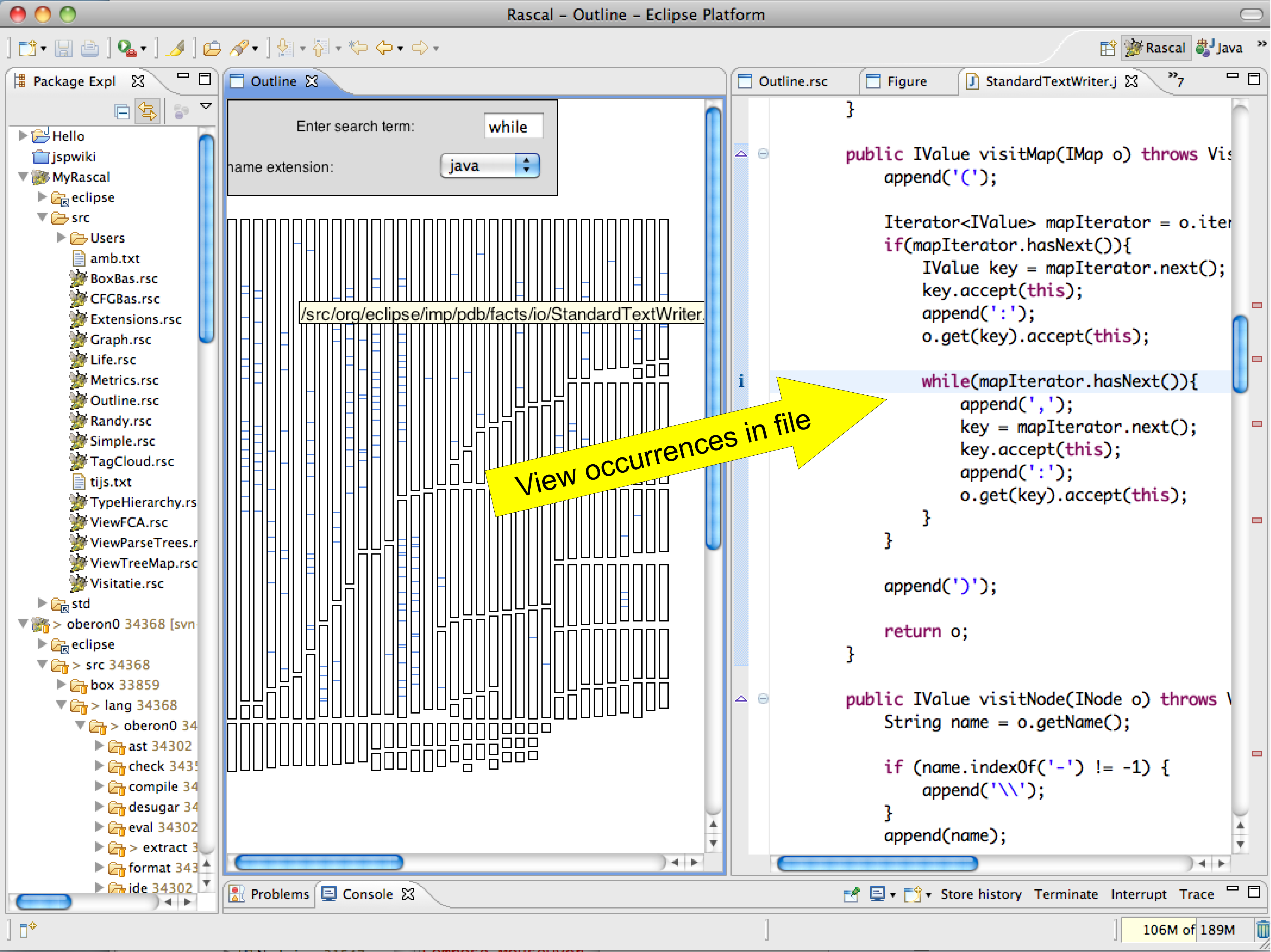
One-stop-shop

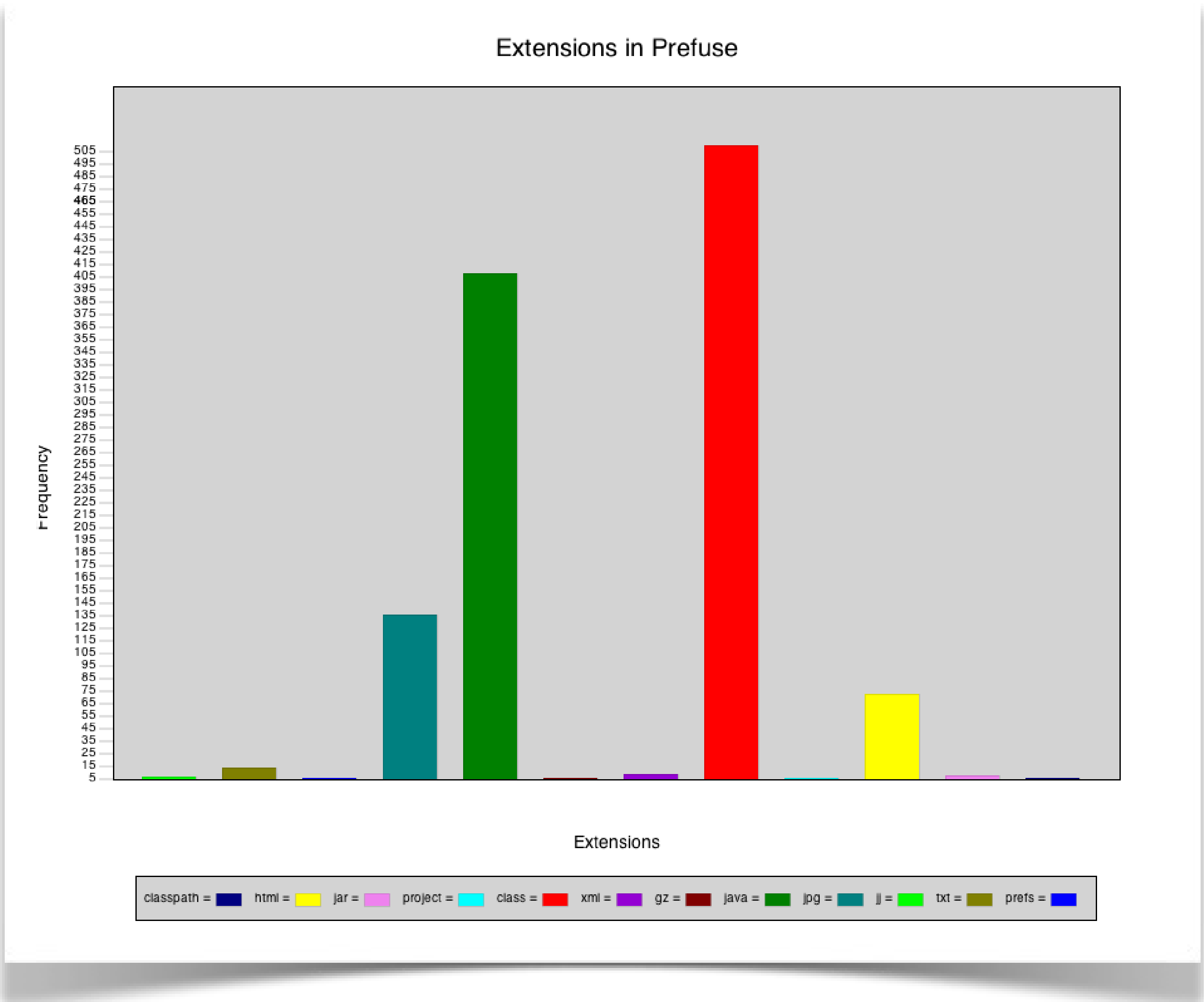
RASCAL

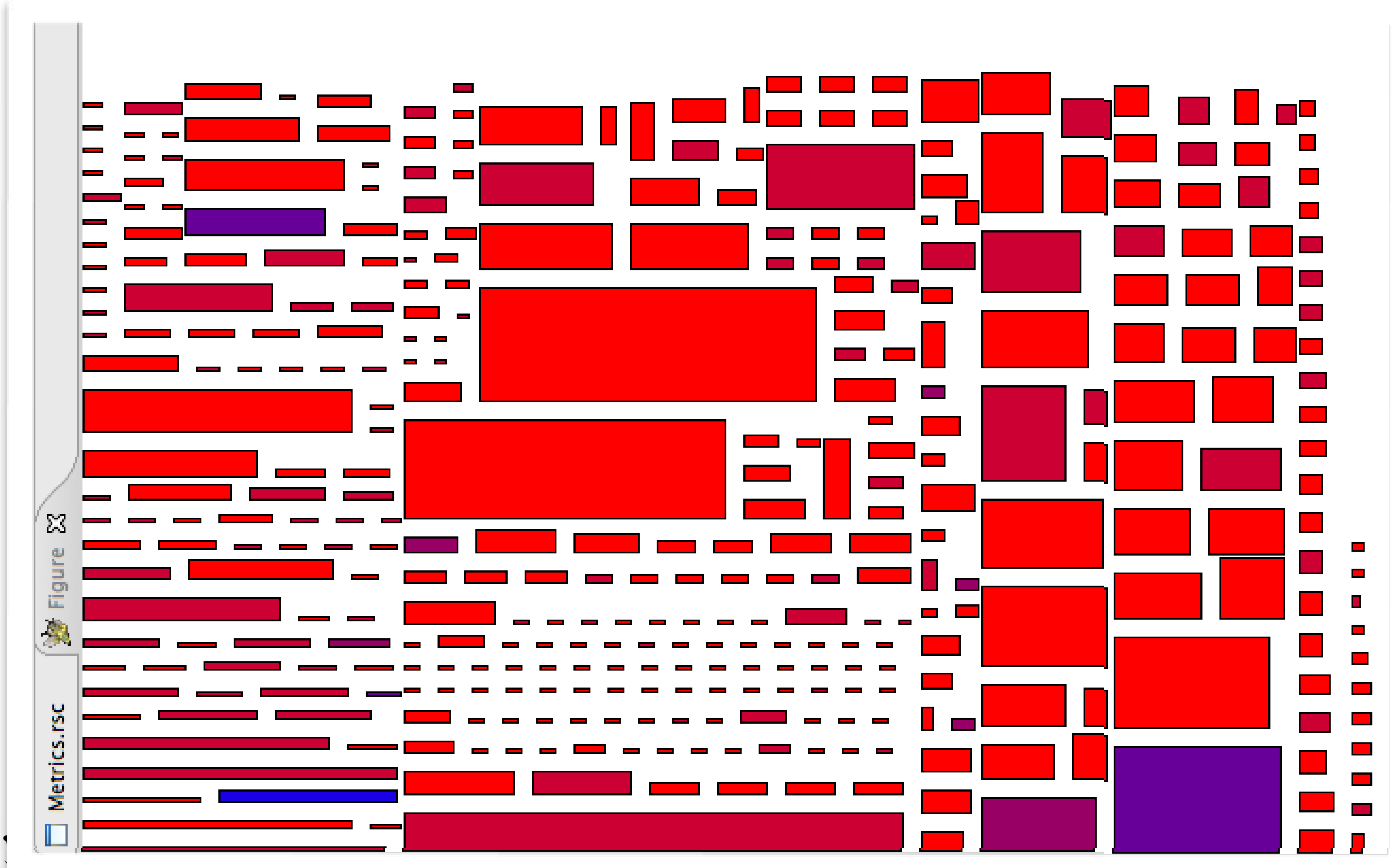
How to integrate Software Visualization in Rascal?



RASCAL


Rascal - Outline - Eclipse Platform
Rascal
Java
Package Expl
Hello
jspwiki
MyRascal
eclipse
src
Users
amb.txt
BoxBas.rsc
CFGBas.rsc
Extensions.rsc
Graph.rsc
Life.rsc
Metrics.rsc
Outline.rsc
Randy.rsc
Simple.rsc
TagCloud.rsc
tijs.txt
TypeHierarchy.rs
ViewFCA.rsc
ViewParseTrees.r
ViewTreeMap.rsc
Visitatie.rsc
std
> oberon0 34368 [svn
eclipse
> src 34368
box 33859
> lang 34368
> oberon0 34
ast 34302
check 343
compile 34
desugar 34
eval 34302
> extract 3
format 343
ide 34302
Outline
Enter search term:
while
name extension:
java
/src/org/eclipse/imp/pdb/facts/io/StandardTextWriter.
View occurrences in file
Outline.rsc
Figure
StandardTextWriter.j
}
public IValue visitMap(IMap o) throws Vis
append('(');
Iterator<IValue> mapIterator = o.iter
if(mapIterator.hasNext()){
IValue key = mapIterator.next();
key.accept(this);
append(':');
o.get(key).accept(this);
while(mapIterator.hasNext()){
append(',');
key = mapIterator.next();
key.accept(this);
append(':');
o.get(key).accept(this);
}
}
append(')');
return o;
}
public IValue visitNode(INode o) throws V
String name = o.getName();
if (name.indexOf('-') != -1) {
append('\\');
}
append(name);
Problems
Console
Store history
Terminate
Interrupt
Trace
106M of 189M

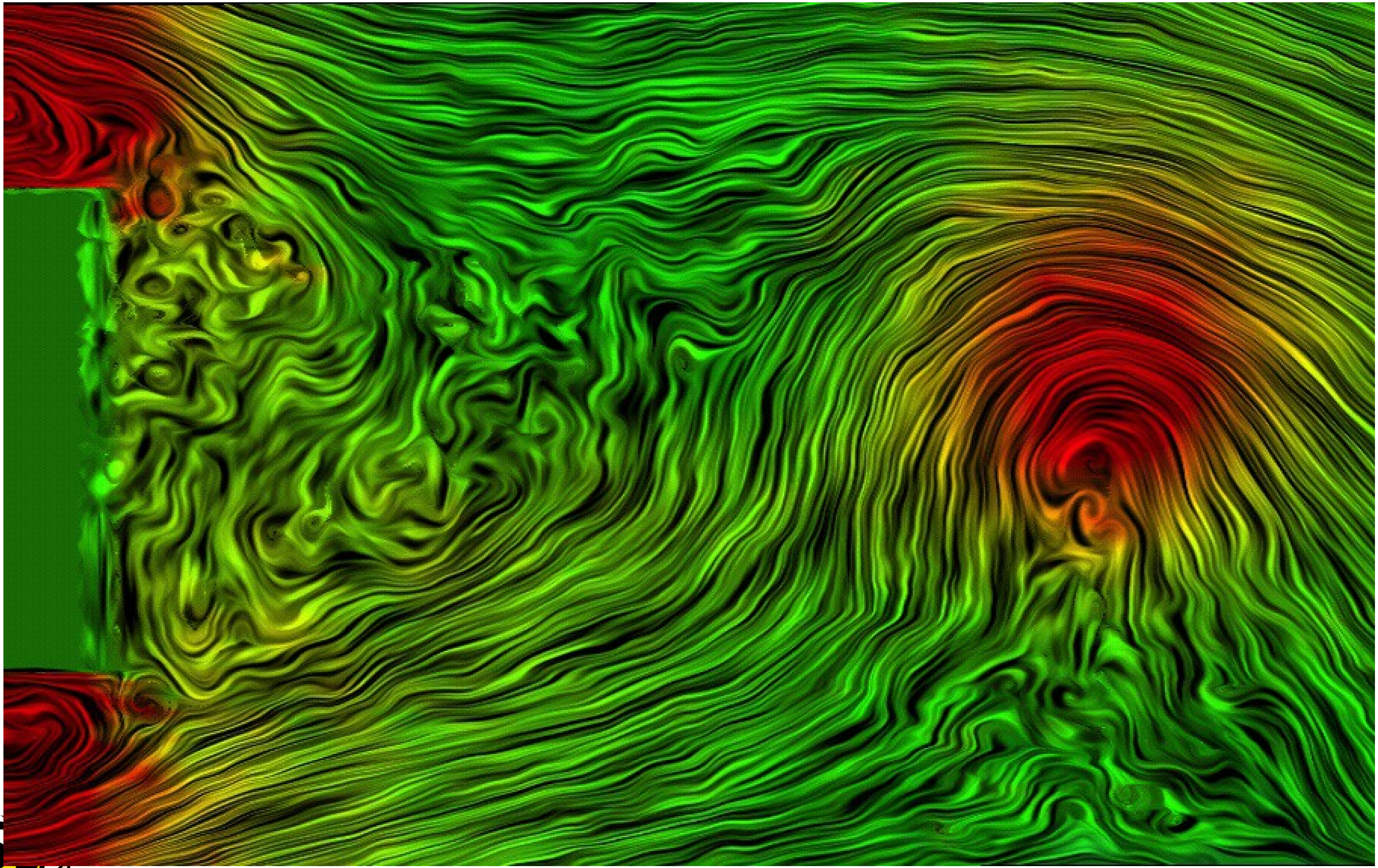

Extensions in Prefuse
Frequency
505
495
485
475
465
455
445
435
425
415
405
395
385
375
365
355
345
335
325
315
305
295
285
275
265
255
245
235
225
215
205
195
185
175
165
155
145
135
125
115
105
95
85
75
65
55
45
35
25
15
5
Extensions
classpath =
html =
jar =
project =
class =
xml =
gz =
java =
jpg =
jj =
txt =
prefs =


Metrics.rsc
Figure

Visual Analytics

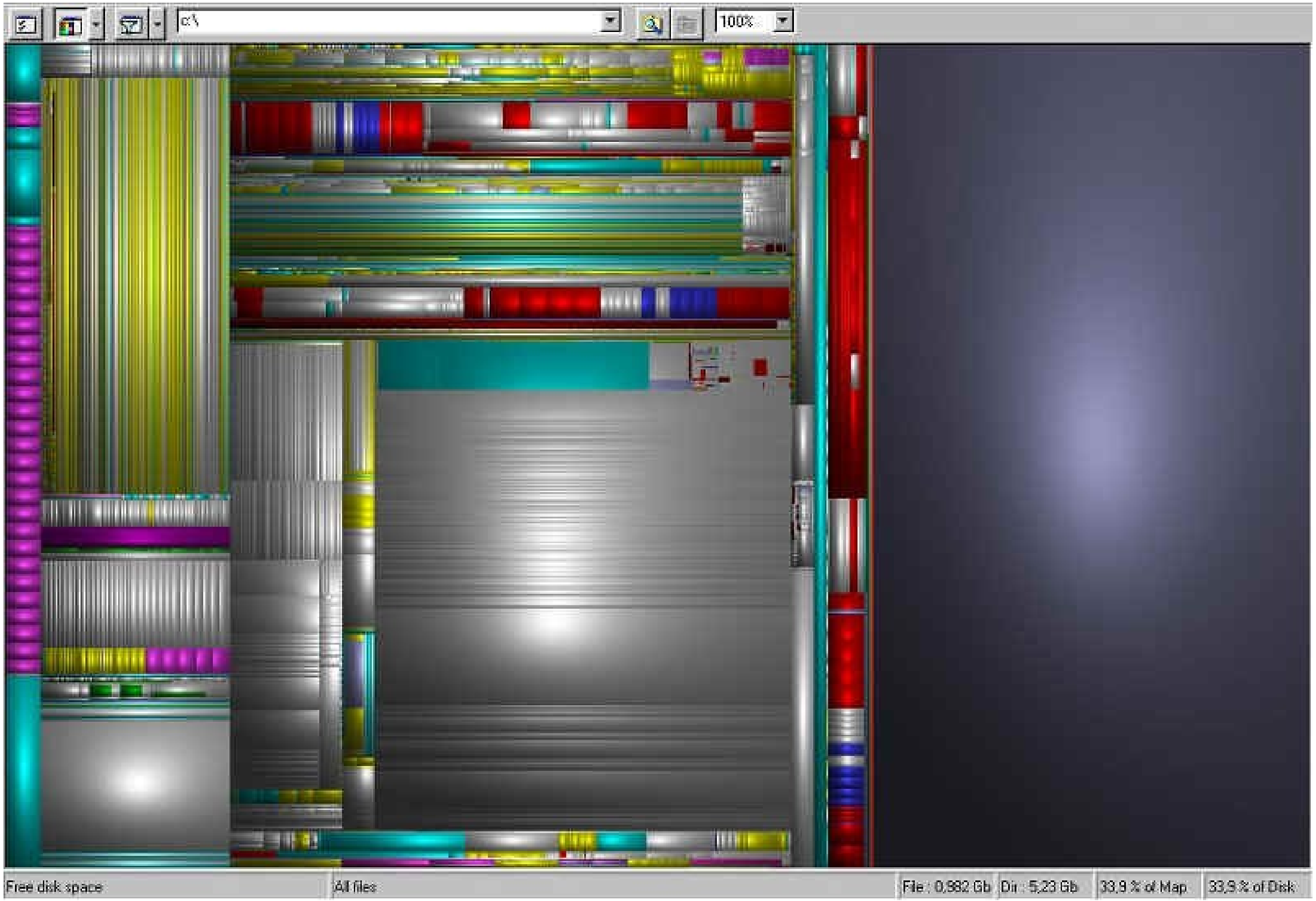
- “*The science of analytical reasoning facilitated by visual interactive interfaces*”
- Related to:
  - Scientific visualization: deals with data that are geometric in nature
  - Information visualization: deals with abstract data structures like trees, graphs and relations
    - Software visualization: deals with software artifacts
  - Visual analytics: aims at reasoning and sense making

Scientific Visualization: *Liquid Flow*



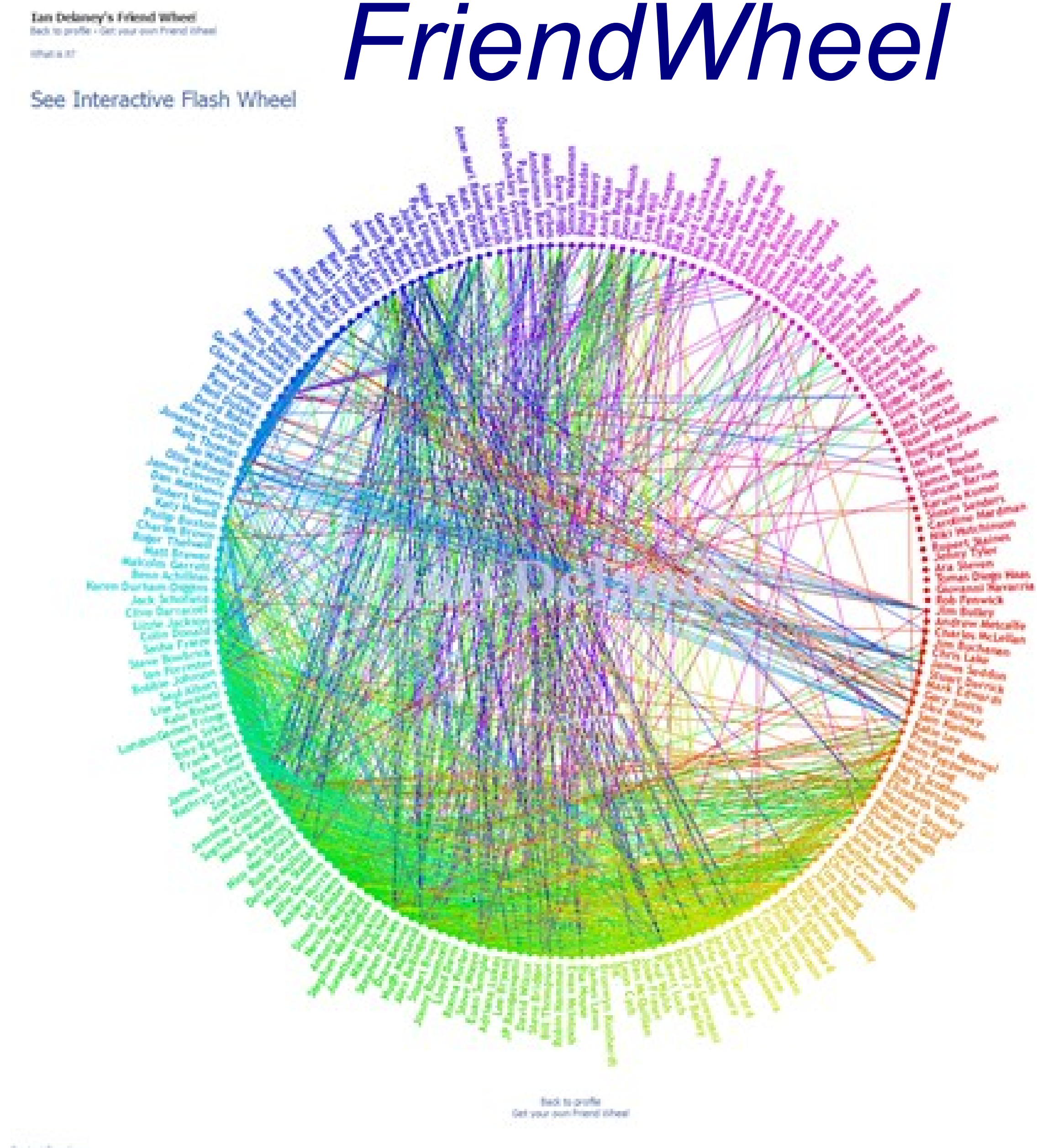
Credit: R. Van Liere & W. De Leeuw, CWI

Information Visualization
Disk Usage

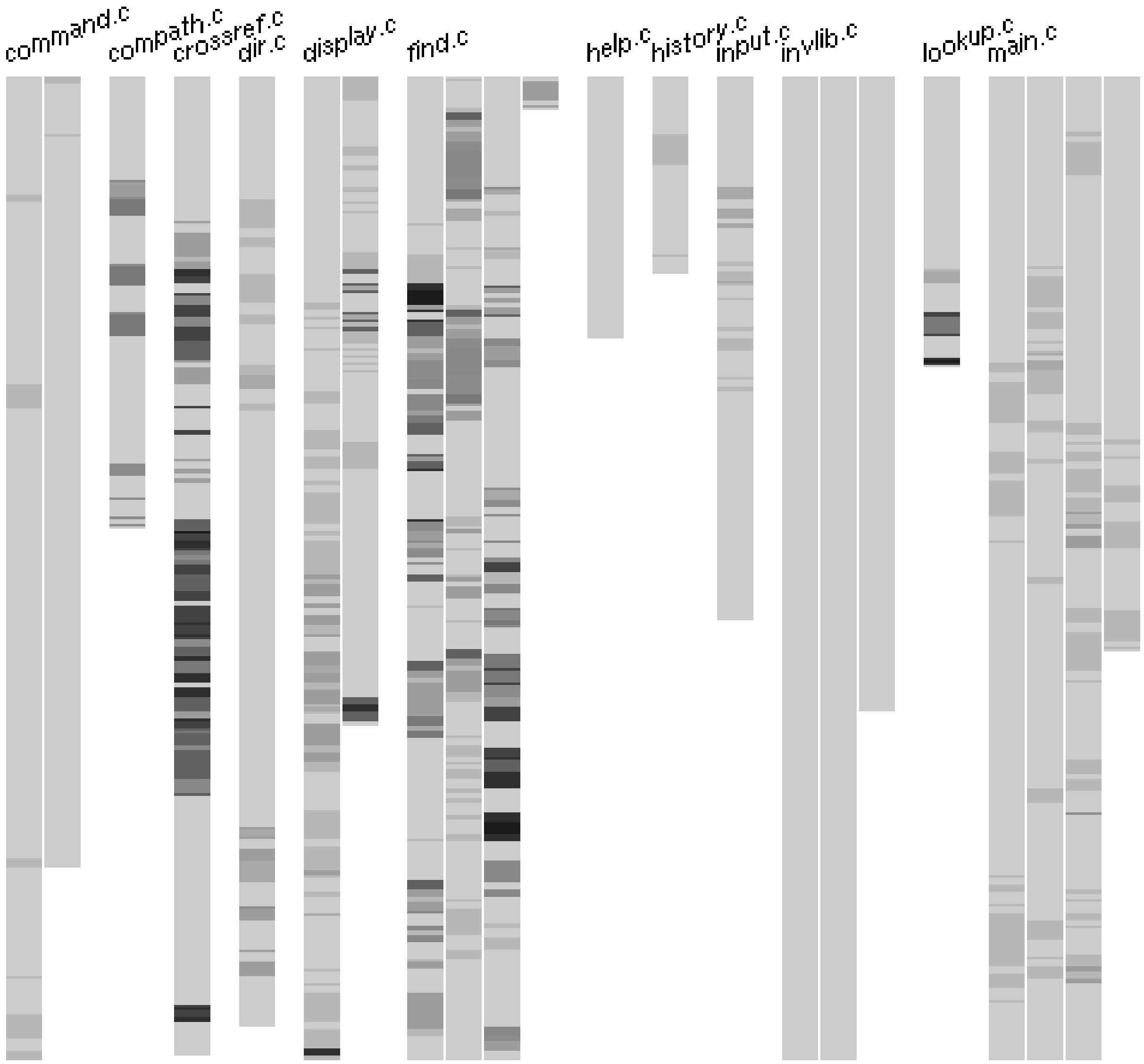


Credits: Jarke van Wijk, TU/e

Information Visualization: *FriendWheel*

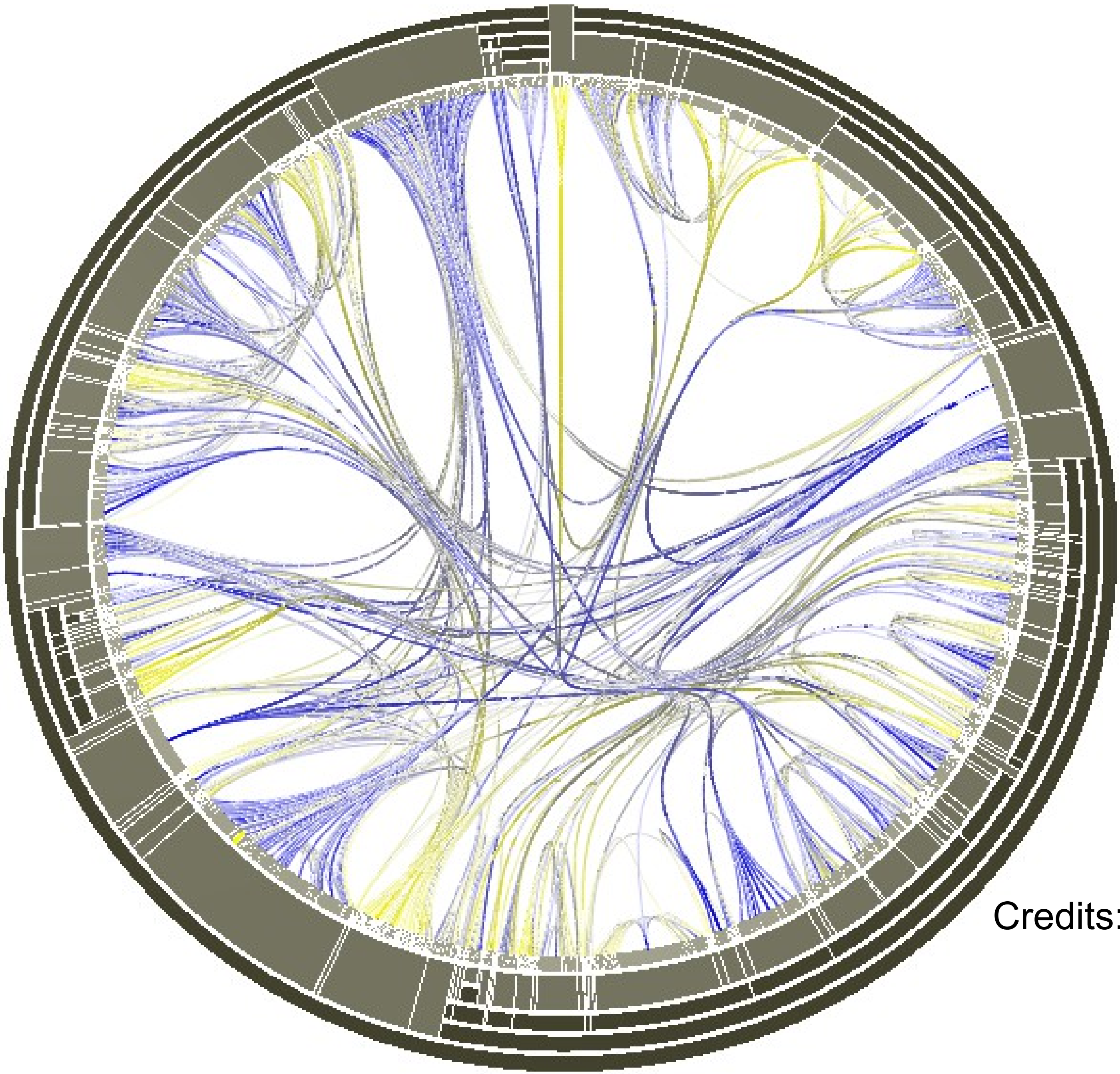


Software Visualisation
Execution Frequency



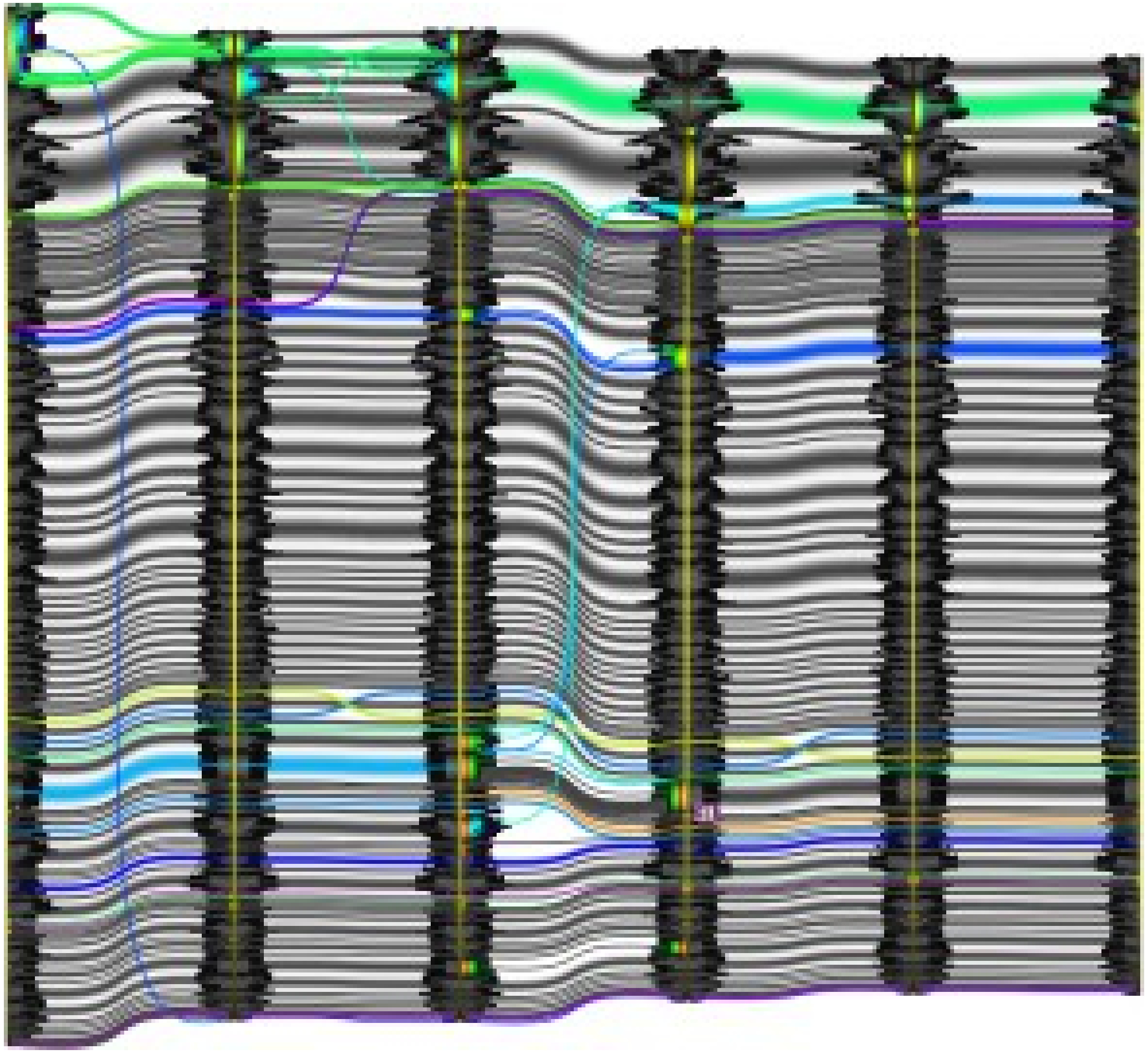
Credits:
Steven Eick

Circular Bundle View



Credits: Daniel Bierwirth

Software Visualisation
Revision histories

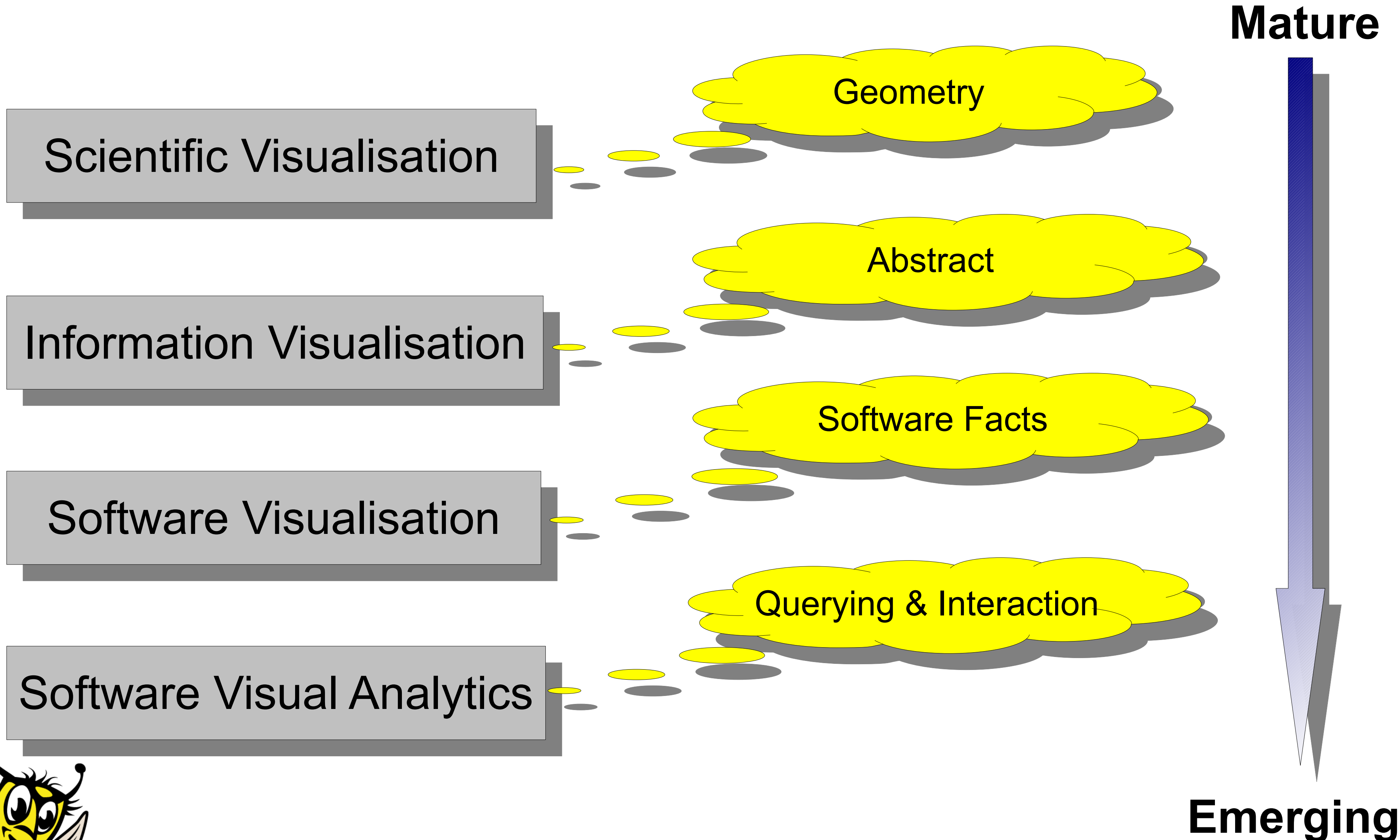


Credit: Alex Telea, RUG

Software Visual Analytics

- Emerging field where data extracted from software artifacts are visualized in order to
  - **Understand** the software
    - Architecture? Component dependencies?
  - **Identify** parts with special properties
    - Most complex? Most revisions? Test coverage?
  - **What if** questions
    - What happens if we adapt this part?

Summary


Mature
Scientific Visualisation
Geometry
Information Visualisation
Abstract
Software Visualisation
Software Facts
Software Visual Analytics
Querying & Interaction
Emerging

General Design Perspectives

- The “DNA” of visualization
- Visual attributes
- Attention
- Polymetric Views
- Tufte's Graphics Design Principles
- Schneiderman's Interaction Mantra

The "DNA" of Visualization

- Most visualizations consist of mappings between data values and visual attributes
- New visualizations can be made by varying these mappings

Visual Attributes

- Position
- Length
- Angle
- Slope
- Area
- Volume
- Density
- Color Saturation
- Color Hue
- Texture
- Connection
- Containment
- Shape

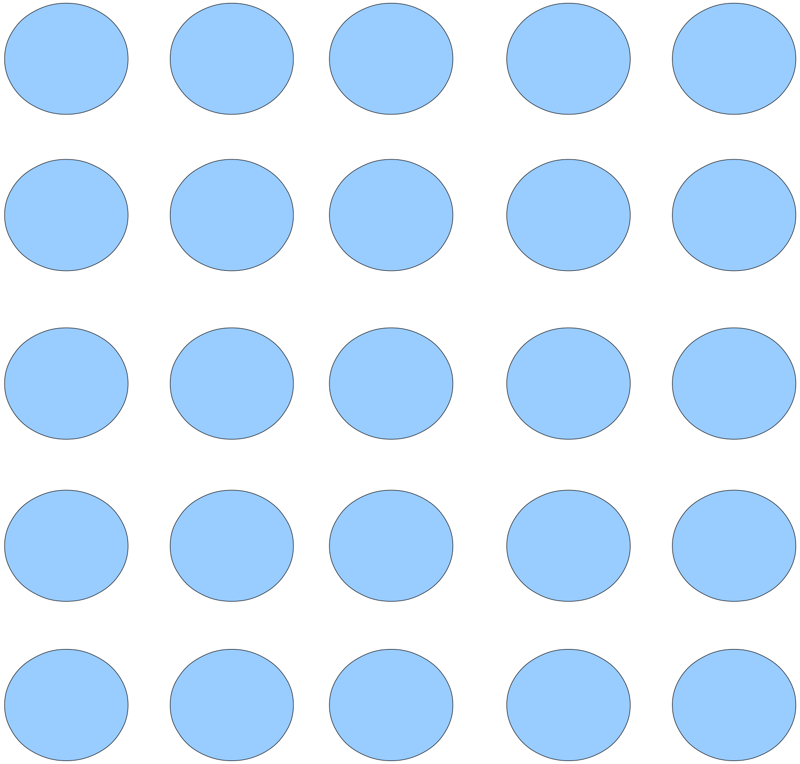
Visual Attributes and Perception

- Different visual attributes have different effect on human visual perception
- Position, size and area are easily seen.
- Changes in color hue or saturation can be difficult to see.

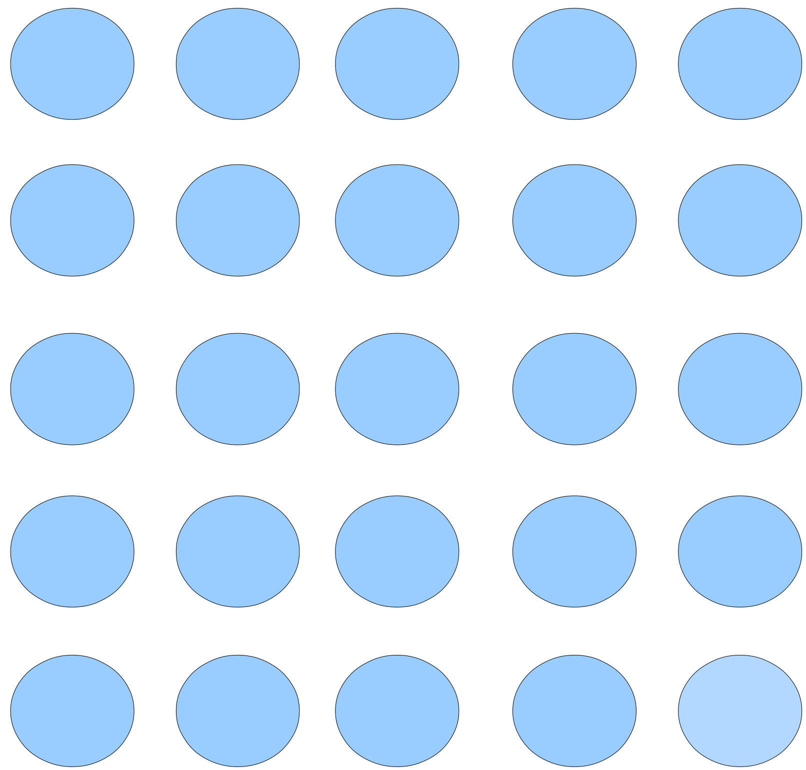
Hue (basic RGB color)

Saturation (colorfulness)

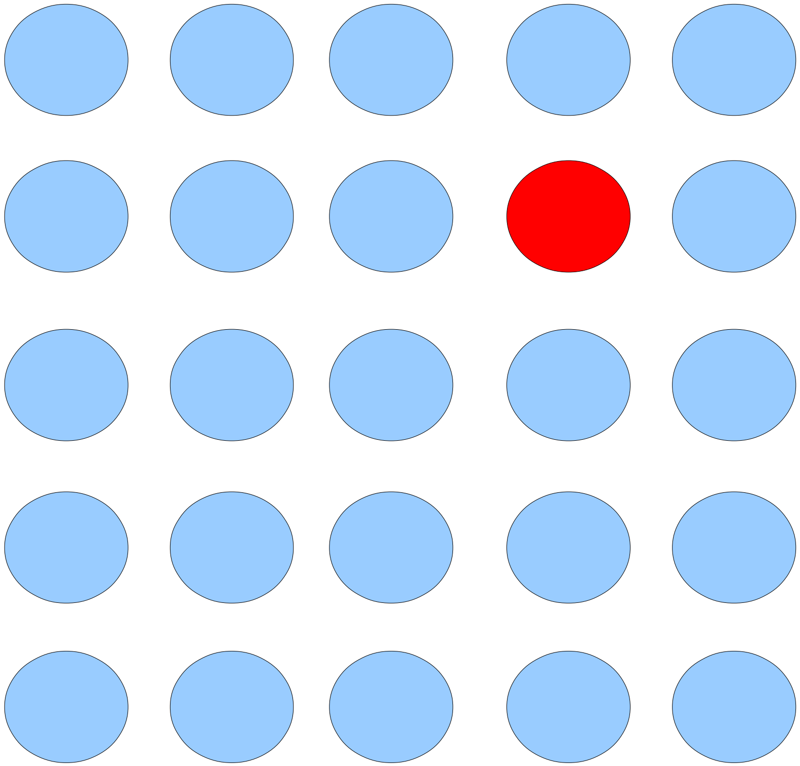
Perception



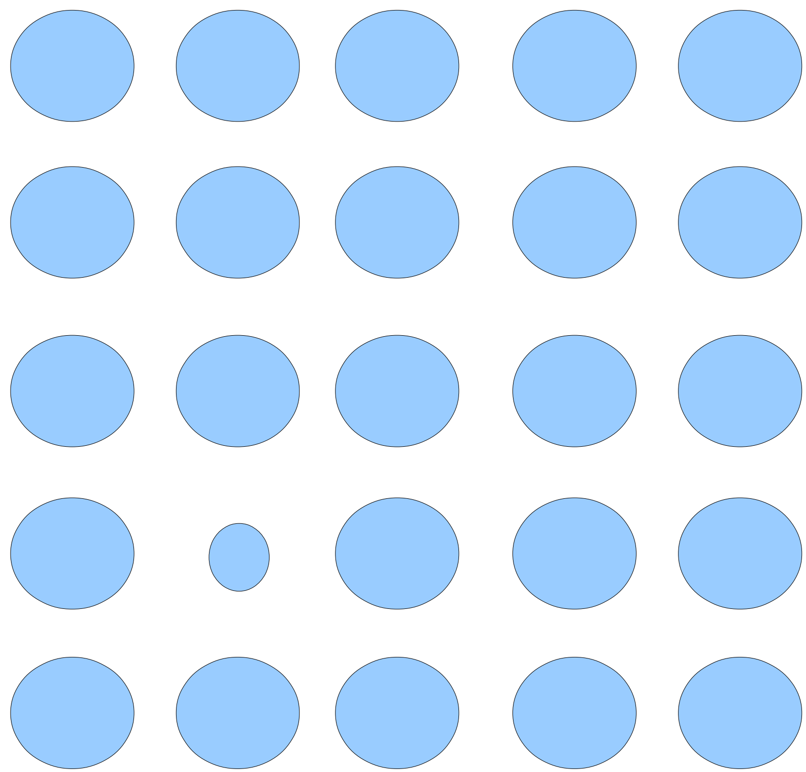
Perception



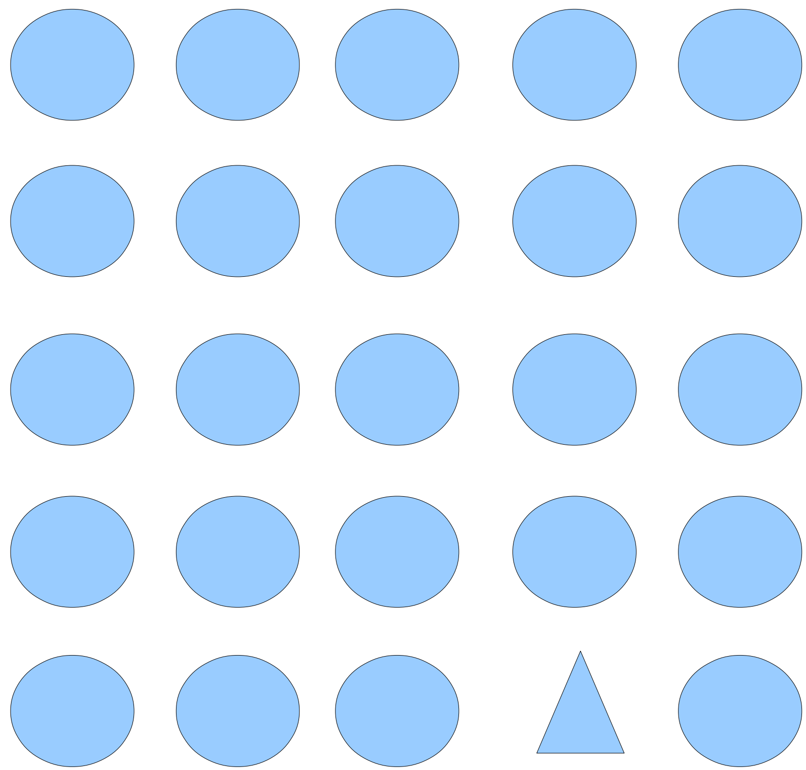
Perception



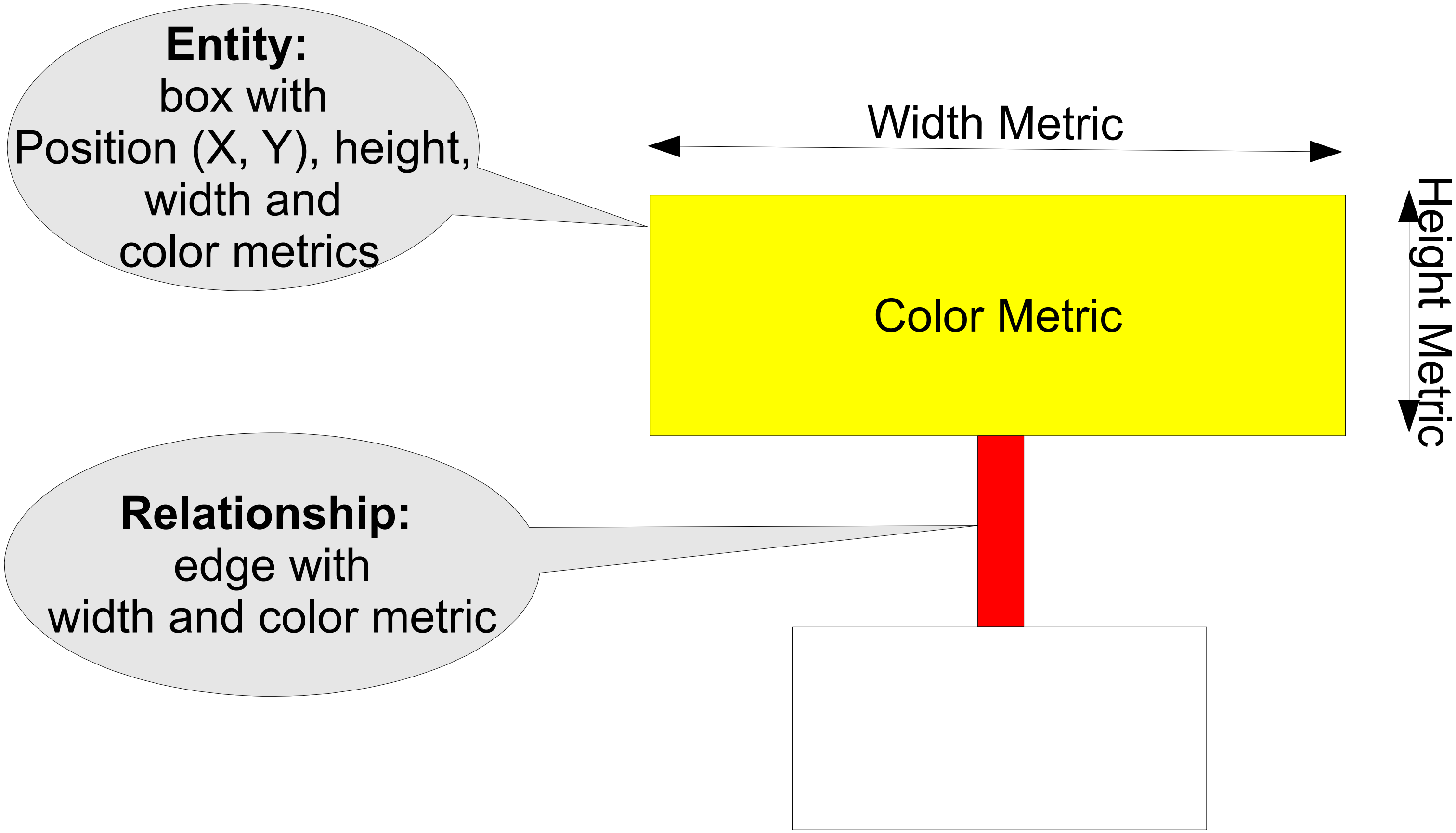
Perception



Perception

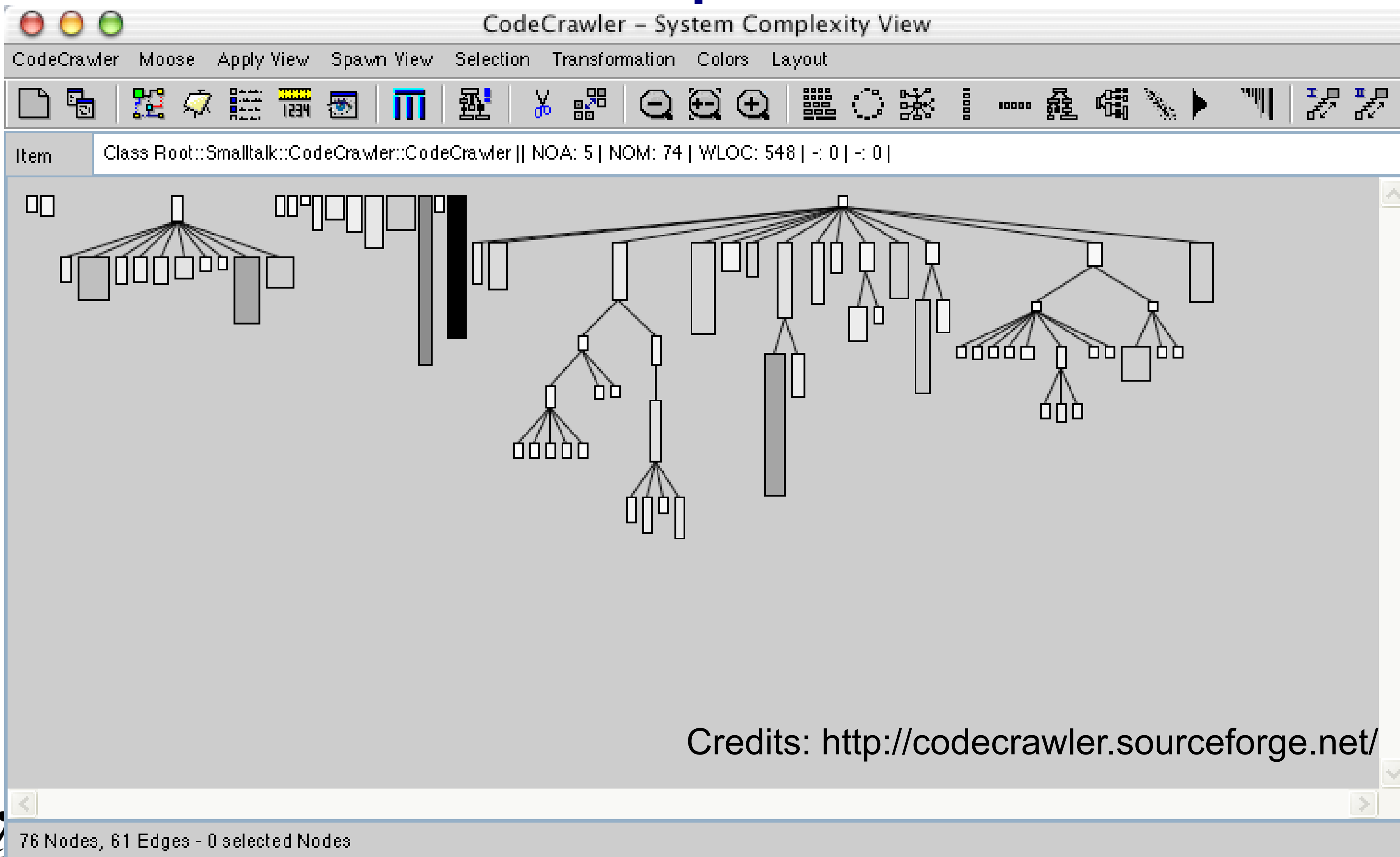


Polymetric Views



M. Lanza & S. Ducasse, CodeCrawler – An Extensible and Language Independent 2D and 3D Software Visualization Tool

Example


CodeCrawler – System Complexity View
CodeCrawler Moose Apply View Spawn View Selection Transformation Colors Layout
Item
Class Root::Smalltalk::CodeCrawler::CodeCrawler || NOA: 5 | NOM: 74 | WLOC: 548 | -: 0 | -: 0 |
Credits: http://codecrawler.sourceforge.net/
76 Nodes, 61 Edges - 0 selected Nodes

Tufte's Graphics Design Principles

- Graphical excellence gives to the viewer
  - the greatest number of ideas
  - in the shortest time
  - with the least ink
  - in the smallest space.
- Maximize data-ink ratio
  - Data-ink ratio = (data ink)/(total ink to print graphic)

Schneiderman's Interaction Mantra

- Overview first
- Zoom and filter
- Details-on-demand

Recall: Software Visual Analytics

- *"The science of analytical reasoning about software artifacts facilitated by visual interactive interfaces"*
- What we want:
  - Integrated software analysis and visualization
- What we already have in Rascal
  - Fact extraction, Fact representation, computation
- What is missing:
  - Visualize all these facts

We do *not* want visualizations that are ...

- *Cluttered* with coordinates, thus:
  - Hard to understand, re-use or compose
- Highly *imperative* and *state-dependent*, thus:
  - Hard to understand, re-use or compose
- Based on too low level representations, thus
  - Hard to understand and automate

```
triangle(10, 10, 10, 200, 45, 200);
rect(45, 45, 35, 35);
quad(105, 10, 120, 10, 120, 200, 80, 200);
ellipse(140, 80, 40, 40);
triangle(160, 10, 195, 200, 160, 200);
```

We want visualizations that are ...

- **Automatic and Domain-specific**
  - Reduce low-level issues (layout, size)
  - Automate mappings (e.g., axis, color scale, ...)
  - Automate interaction support
- **Reusable**
  - Treat figures and visual attributes as ordinary values; can be parameters/result of functions
  - Arbitrary nesting of figures
  - Well-defined composition of visual attributes

We want visualizations that are ...

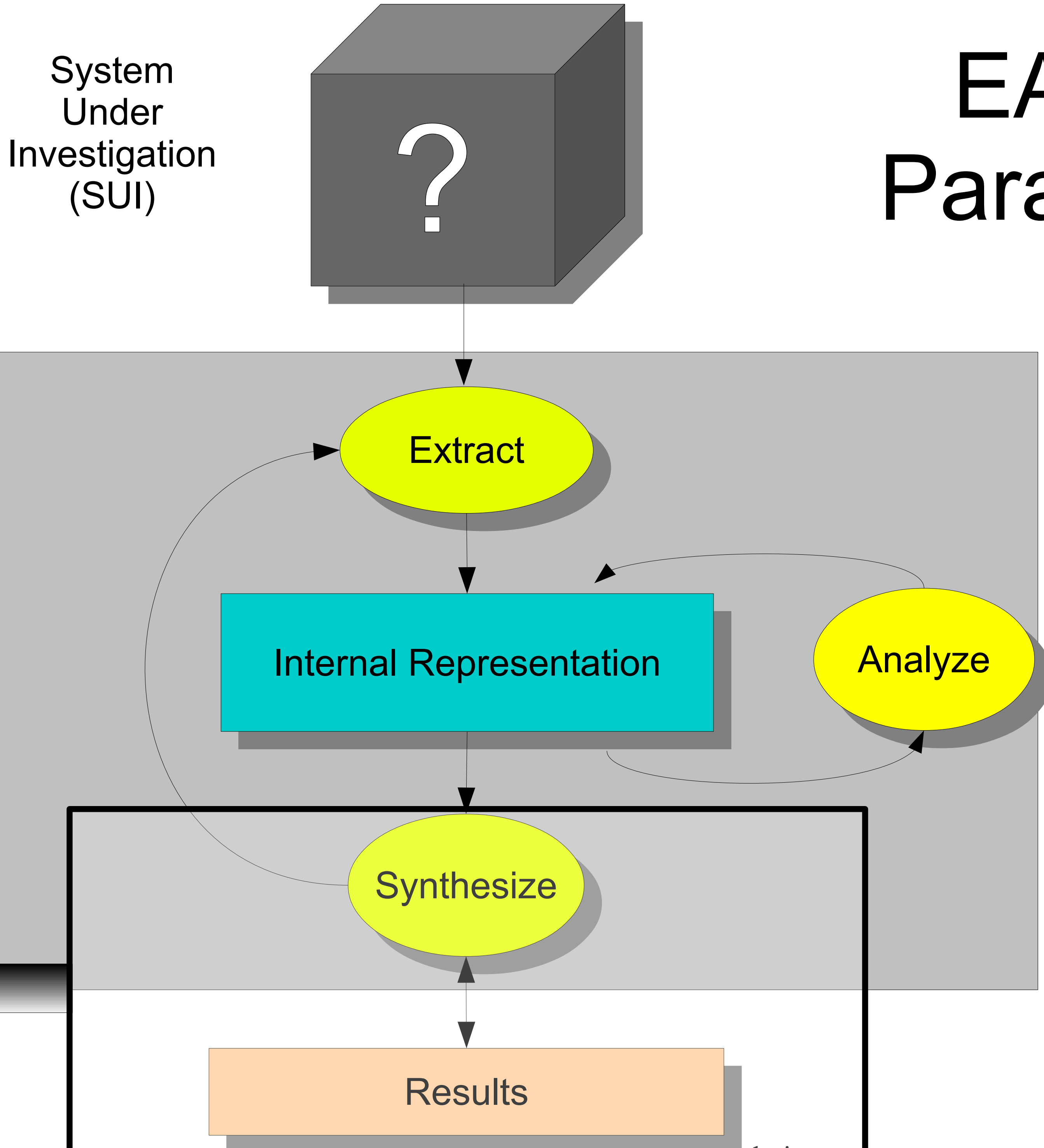
- **Compositional**
  - Global visualization state (e.g. Pen color) hinders composition
  - Self-contained, composable, visualizations
- **Interactive**
  - Enable Schneidermann's Mantra: *Overview First, Zoom and Filter, then Details-on-demand*
  - Provide the GUI-elements (buttons, text fields, ...) to achieve this.

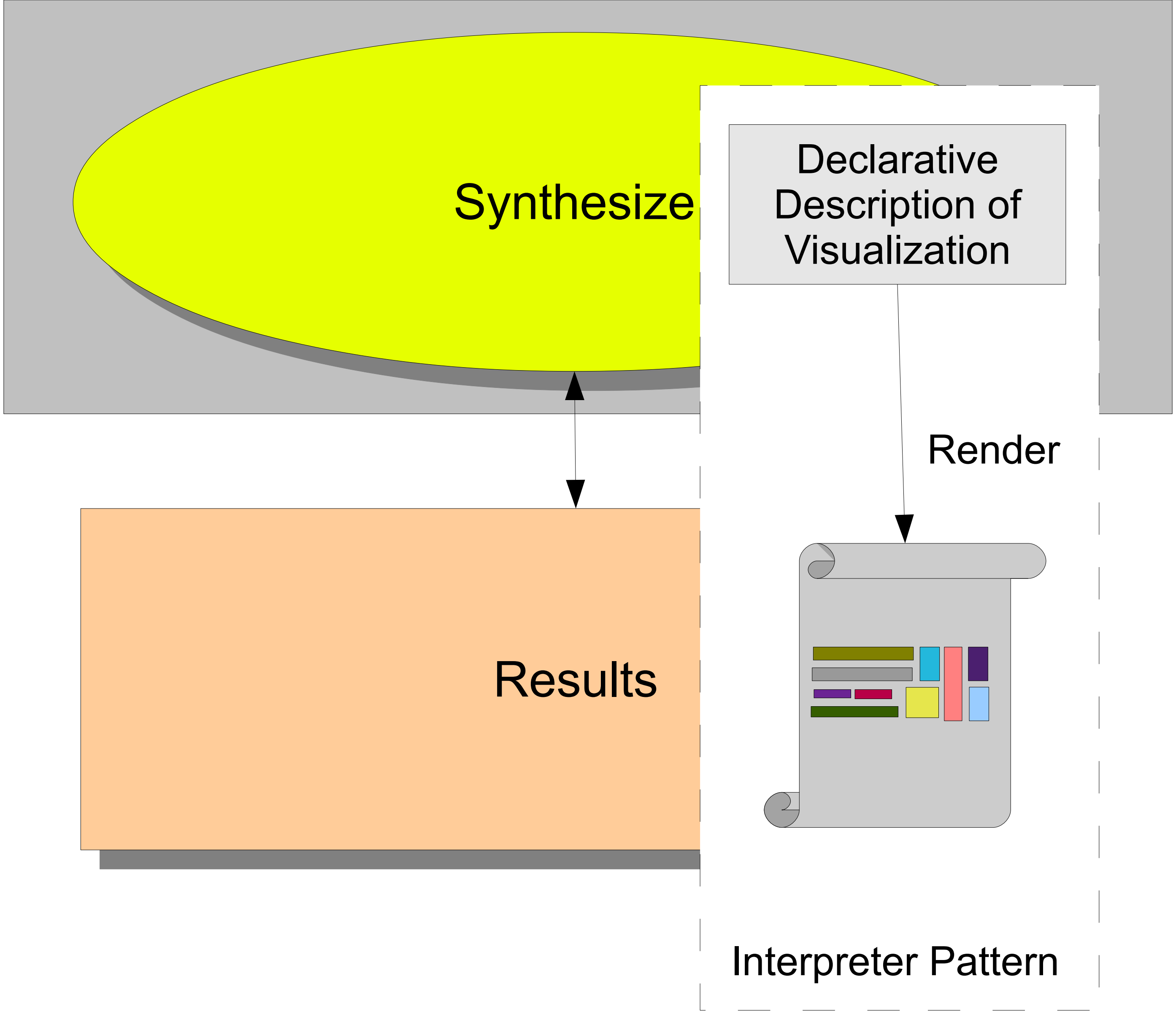
The vis::Figure Library

- A Rascal library to meet these design goals.
- Key words:
  - **Coordinate free**
  - **Compositional & orthogonal**
  - **Interactive**

EASY Paradigm

Remember?



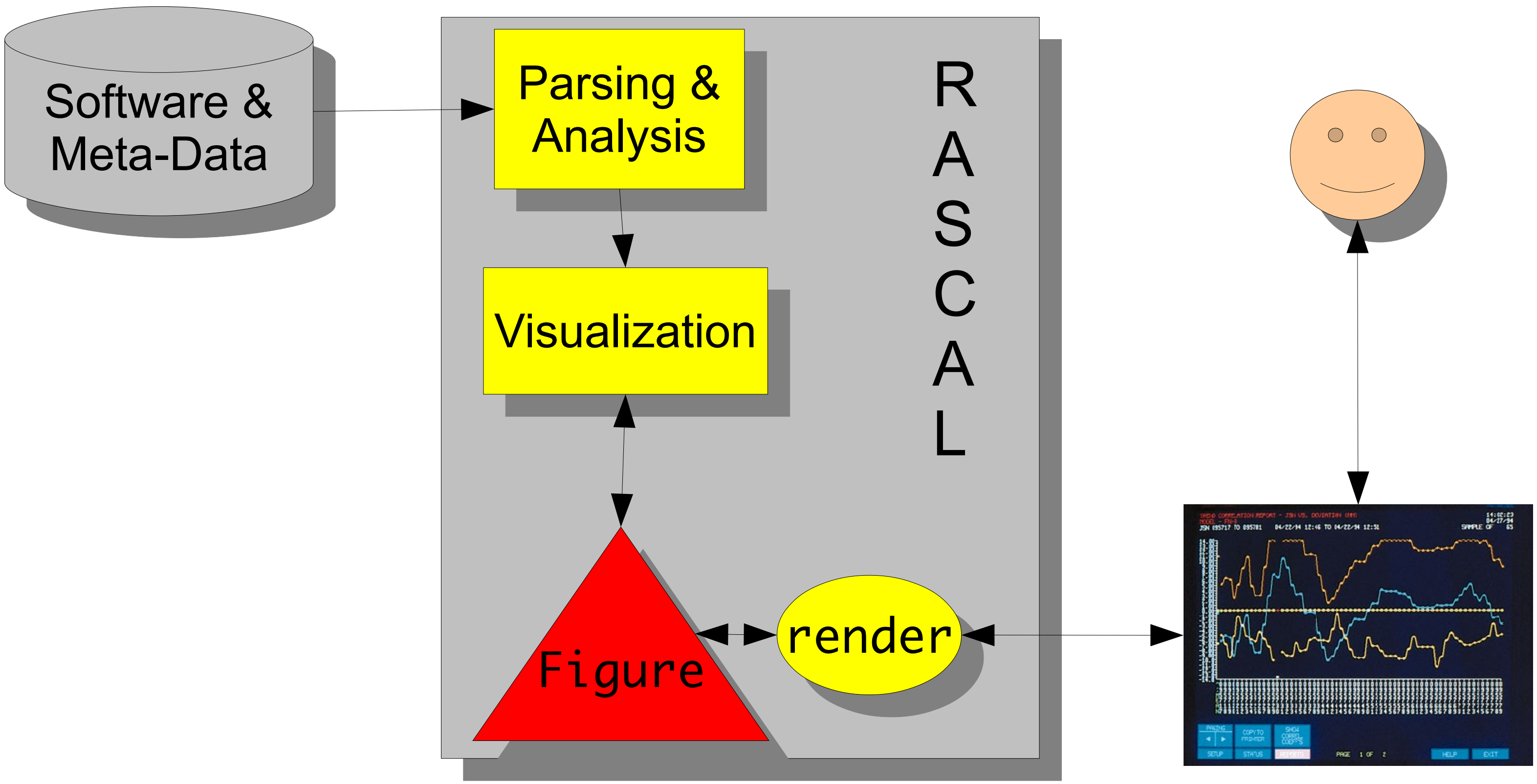


Data types:

Figure
Fproperty

Interpreter:
render

More Technical View



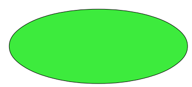
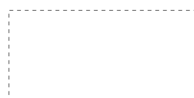
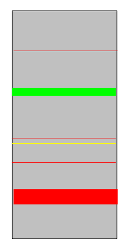
Figures

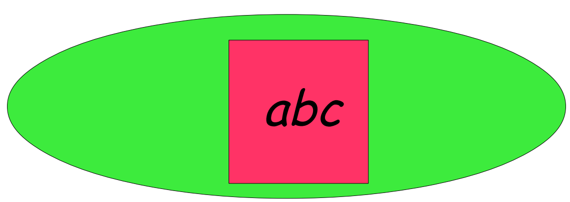
- Figures
  - are described by data type `Figure`
  - are values that can be computed and manipulated (e.g. by Rascal functions)
  - only describe their own properties: dimensions, alignment, color, ...
  - are unaware of their actual coordinates
  - can be composed with other figures: horizontal composition, placement on grid or in tree, graph, ...
  - can be reused in different contexts

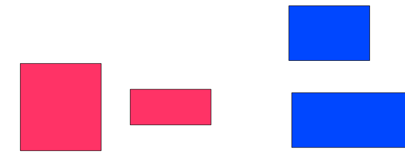
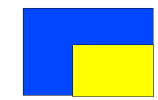
Figure

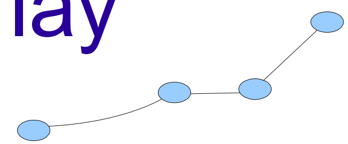
- Primitive figures:
  - text, outline
- (Nested) container figures:
  - box, ellipse, space
- Figure composition operators:
  - hcat, vcat, hvcat, overlay
  - pack, grid, graph, tree

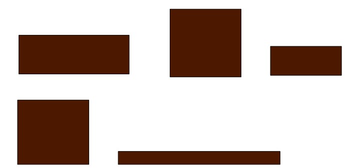
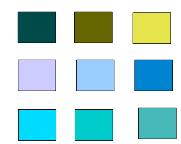
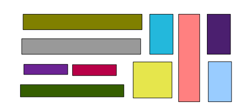
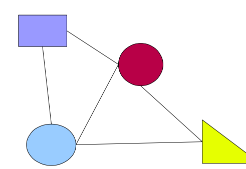
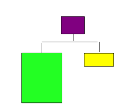
Figure

- Text *abc*
- Box 
- Ellipse 
- Space 
- Outline 
- *Fully nested*



- Hcat, Vcat 
- Overlay 



- HVcat 
- Grid 
- Pack 
- Graph 
- Tree 

FProperty

- **Identity**: id
- **Size**: grow, shrink, resizable
- **Absolute size**: size, width, height, *gap*
- **Alignment**: align, left, right,top, bottom
- **Color**: fillColor, lineColor, fontColor
- **Interaction**: mouseOver, onClick, button, textfield, combobox, computeFigure

FProperty

- All properties have a standard value
- A figure can
  - redefine the value of a property for itself, e.g. `fillColor(“yellow”)`
  - redefine the value of a property for all its children, e.g. `std(fillColor(“yellow”))`
- This inheritance-*like* model
  - promotes re-use
  - allows per figure redefinition of property
  - minimizes number of property settings

Summary: Figures and Properties

- `Figure` and `FProperty` are ordinary user-defined Rascal data types
- We can use the full power of Rascal to
  - Write functions that return `Figure`/`FProperty` values
  - Integrate parsing, fact extraction and visualization
  - Contrast with. e.g., GnuPlot scripting language

Rendering

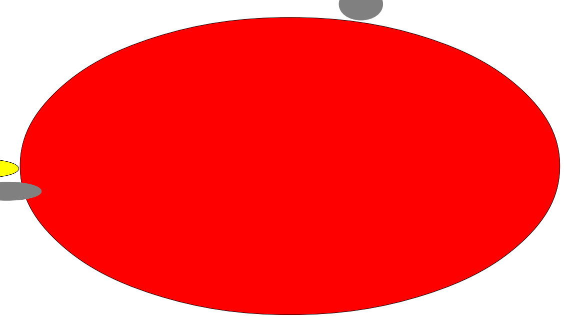
Figure

list[FProperty]

```
render(
    ellipse(fillColor("red"))
)
```

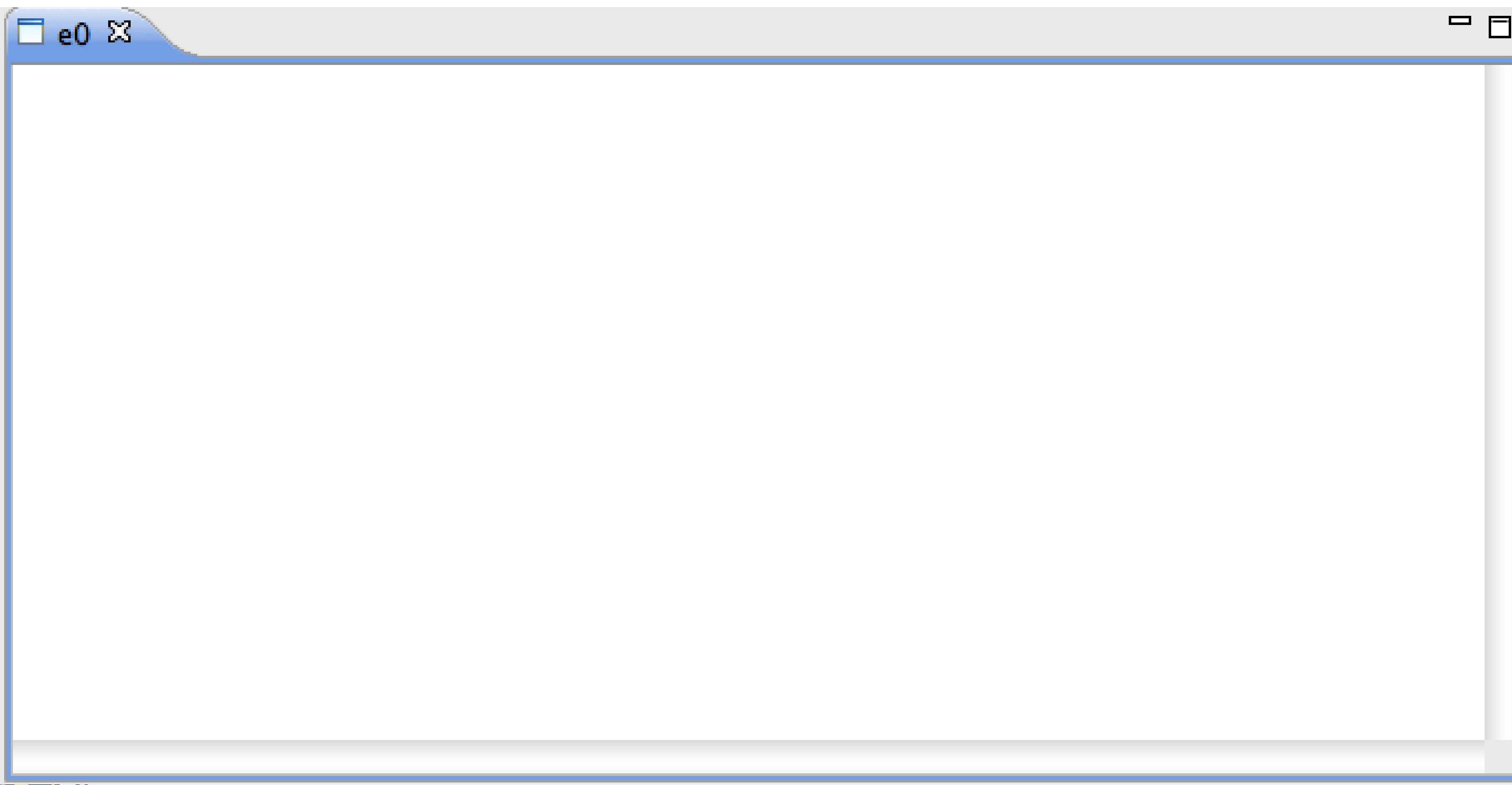
What size will it be?

Where will it appear?

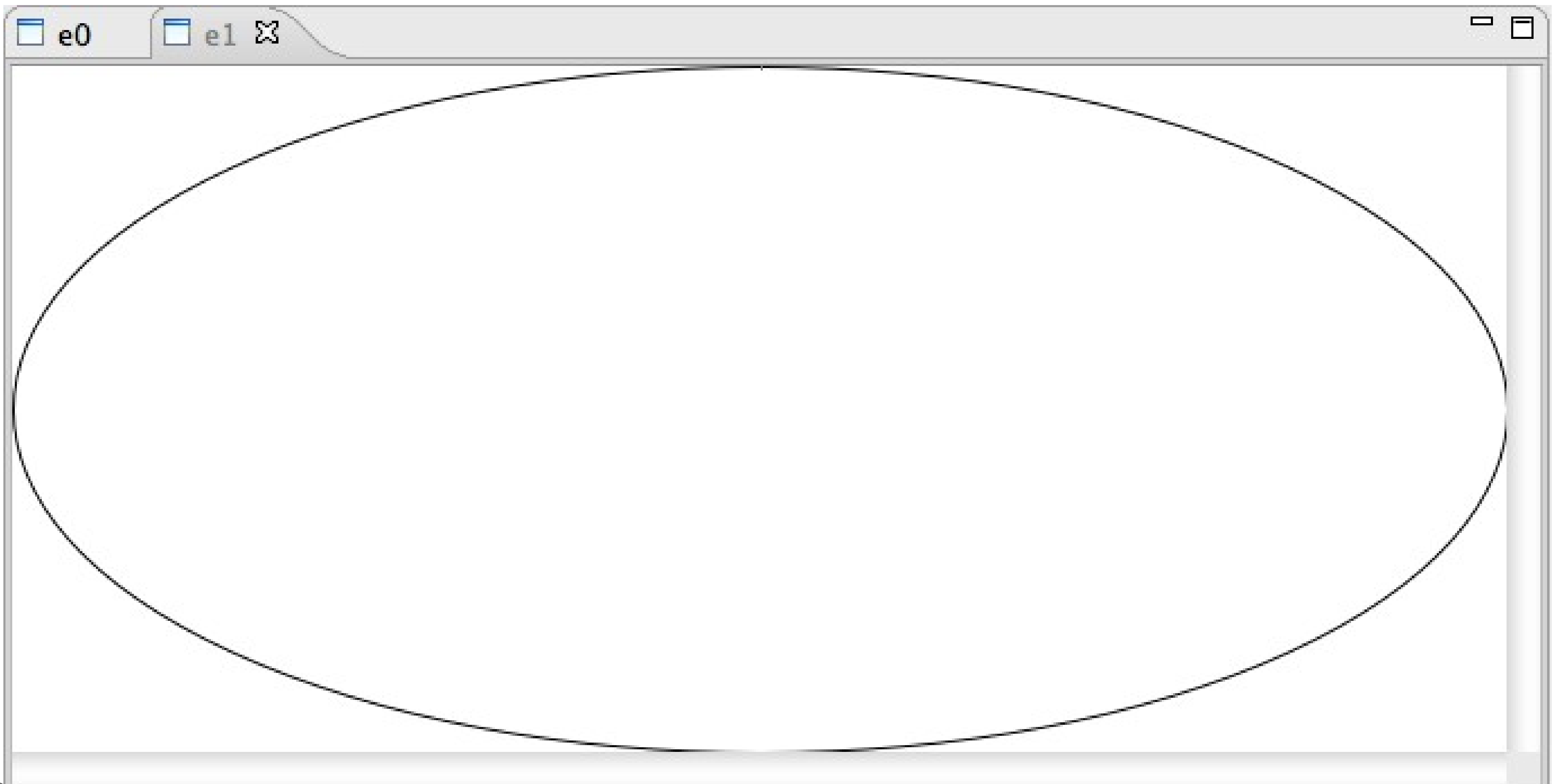


Coordinate-free

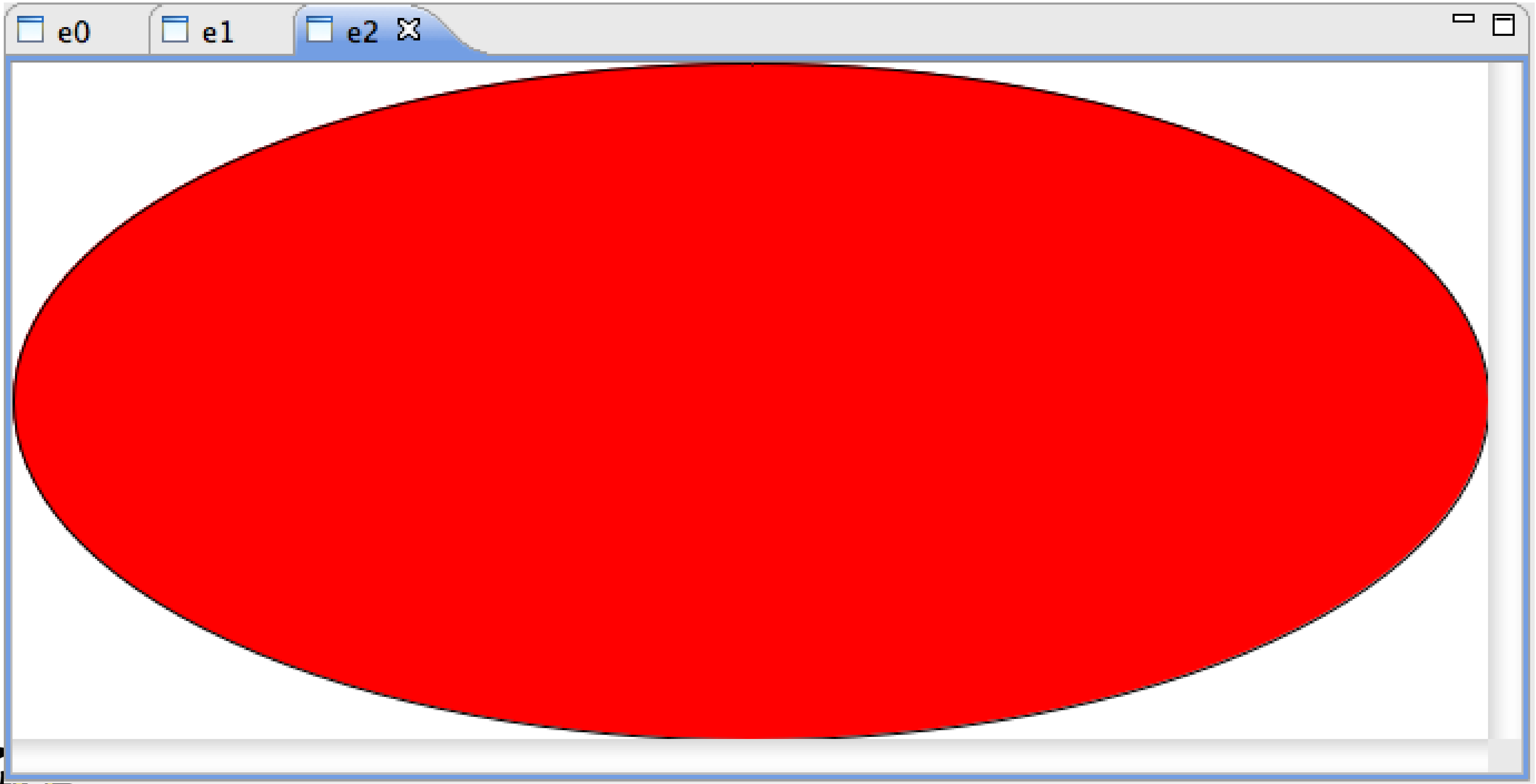
An empty pane ...



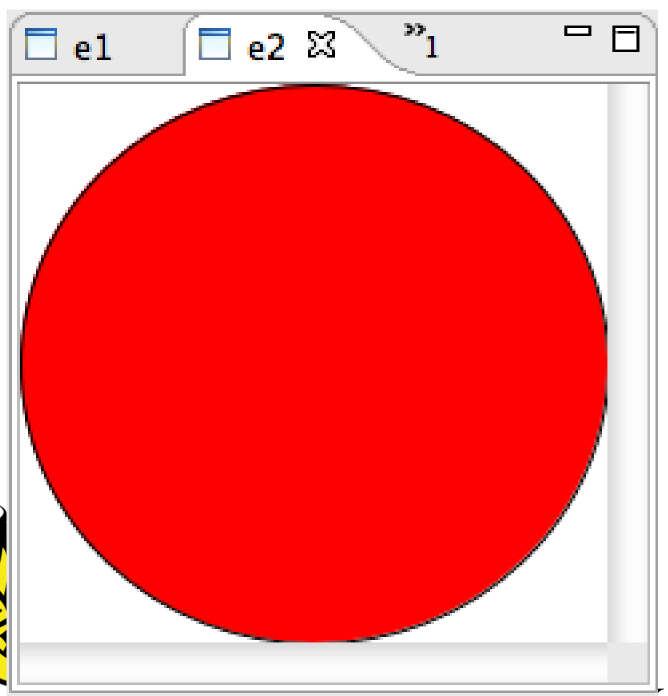
ellipse()

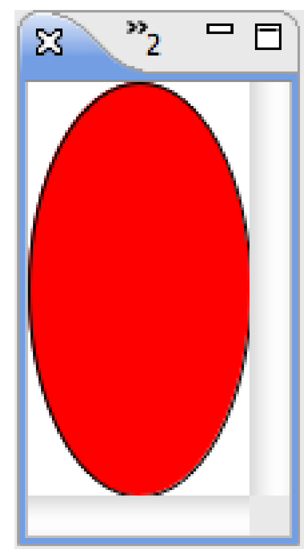


```
ellipse(fillColor("red"))
```

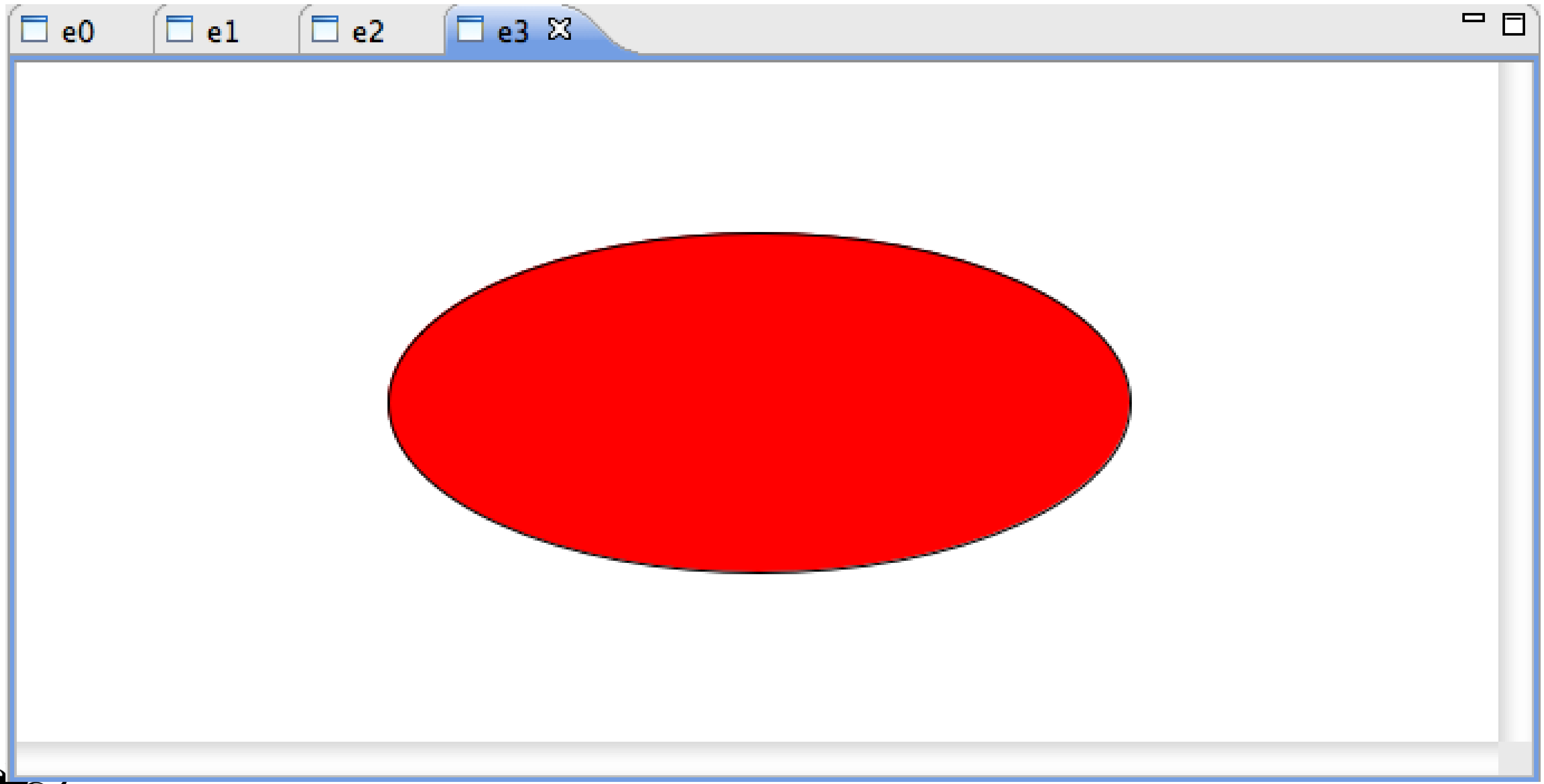


Size is adjusted to available space (turn off per figure with `resizable`)

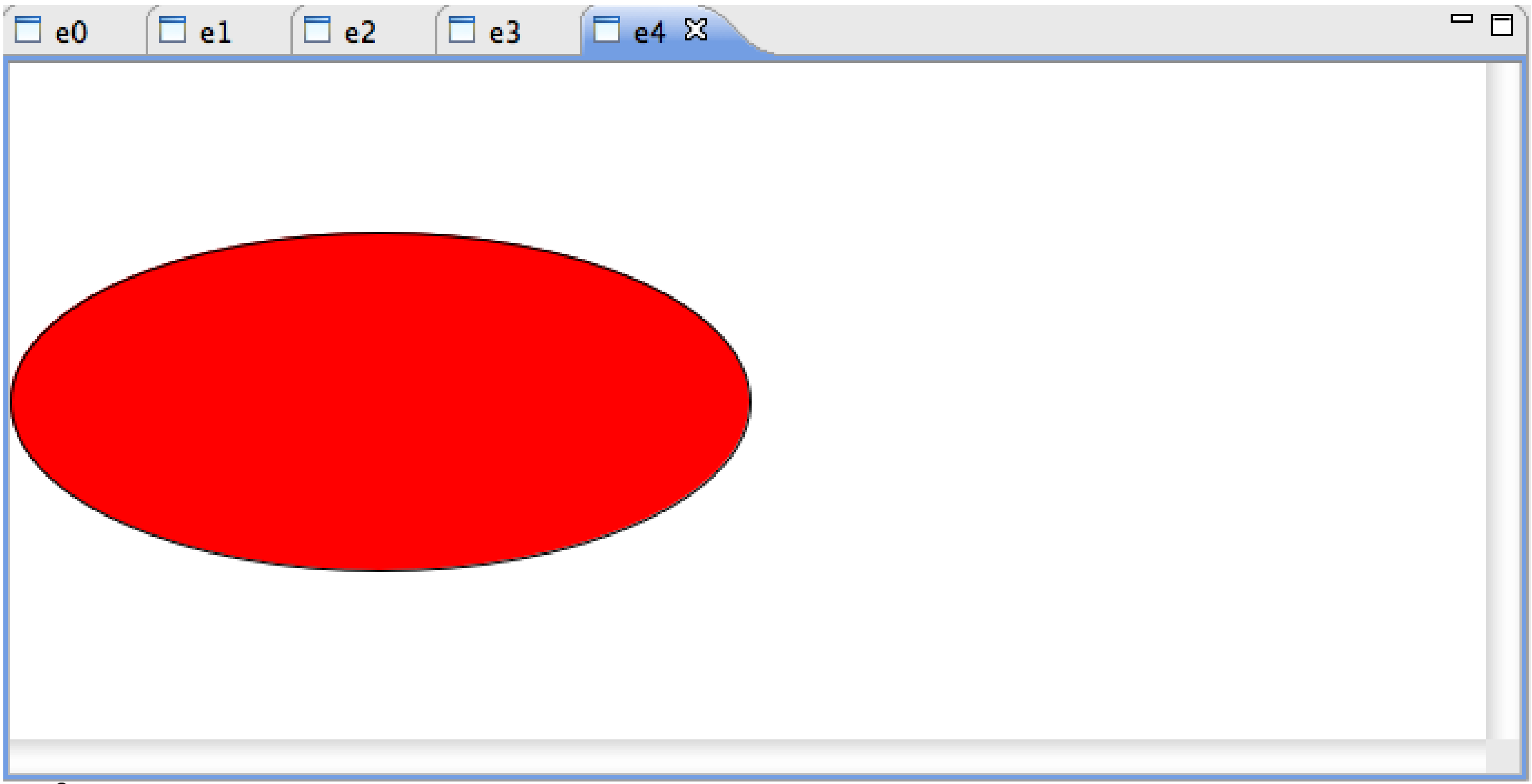




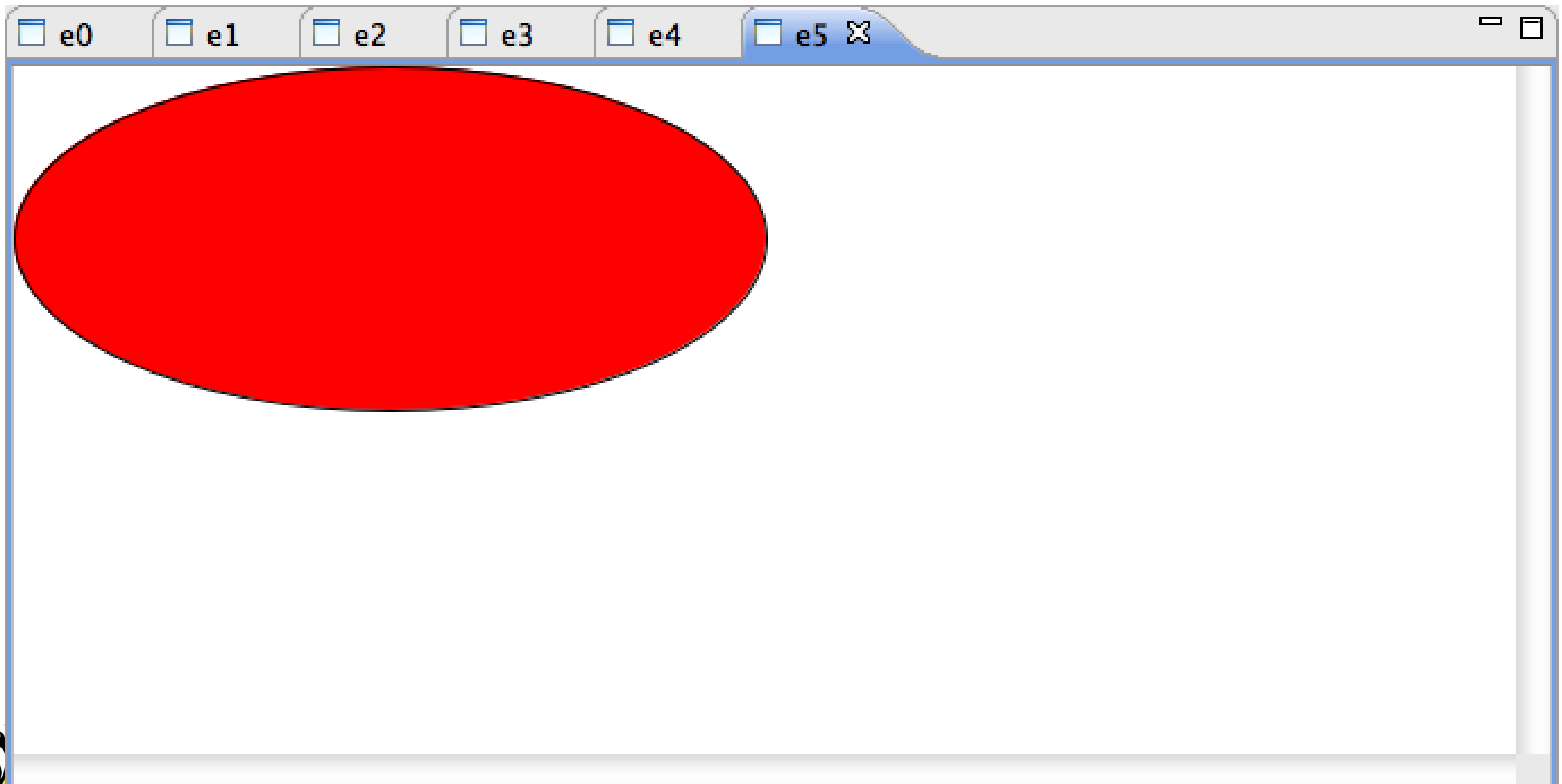
```
ellipse(fillColor("red"), shrink(0.5))
```



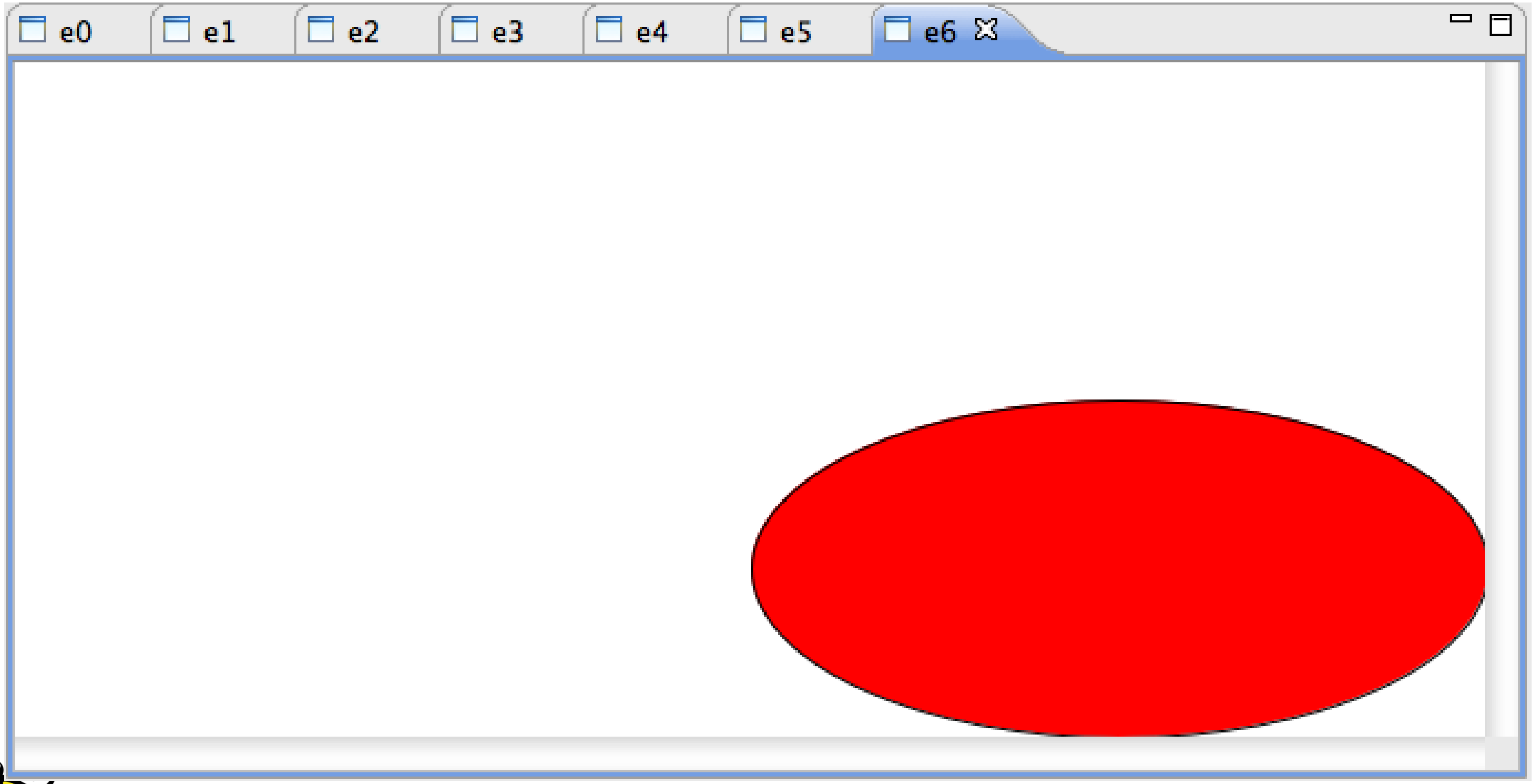
```
ellipse(fillColor("red"), shrink(0.5),
left())
```



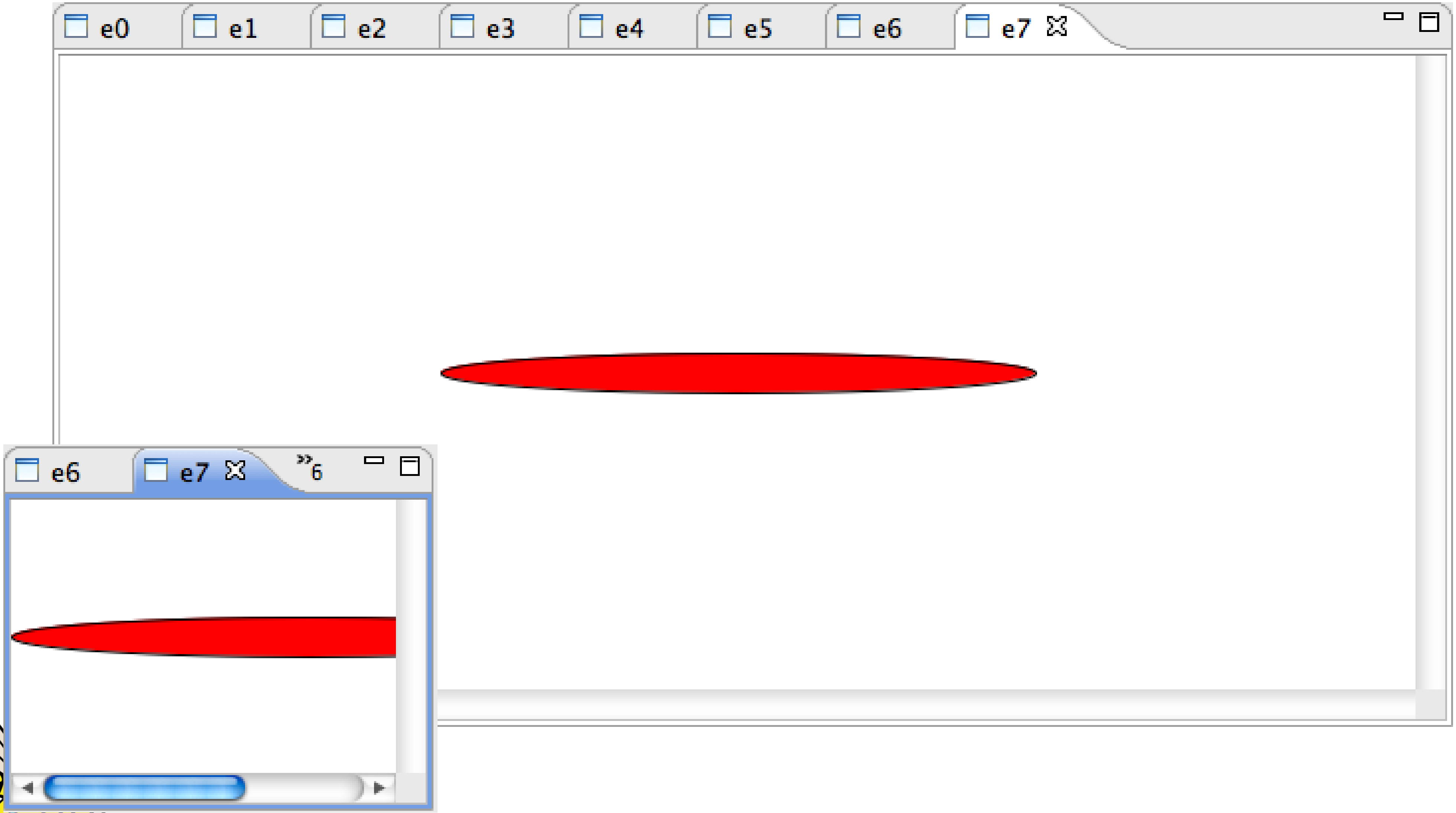
```
ellipse(fillColor("red"), shrink(0.5),
        left(), top())
```



```
ellipse(fillColor("red"), shrink(0.5),
        right(), bottom())
```



```
ellipse(fillColor("red"), size(300, 20))
```



Exercise

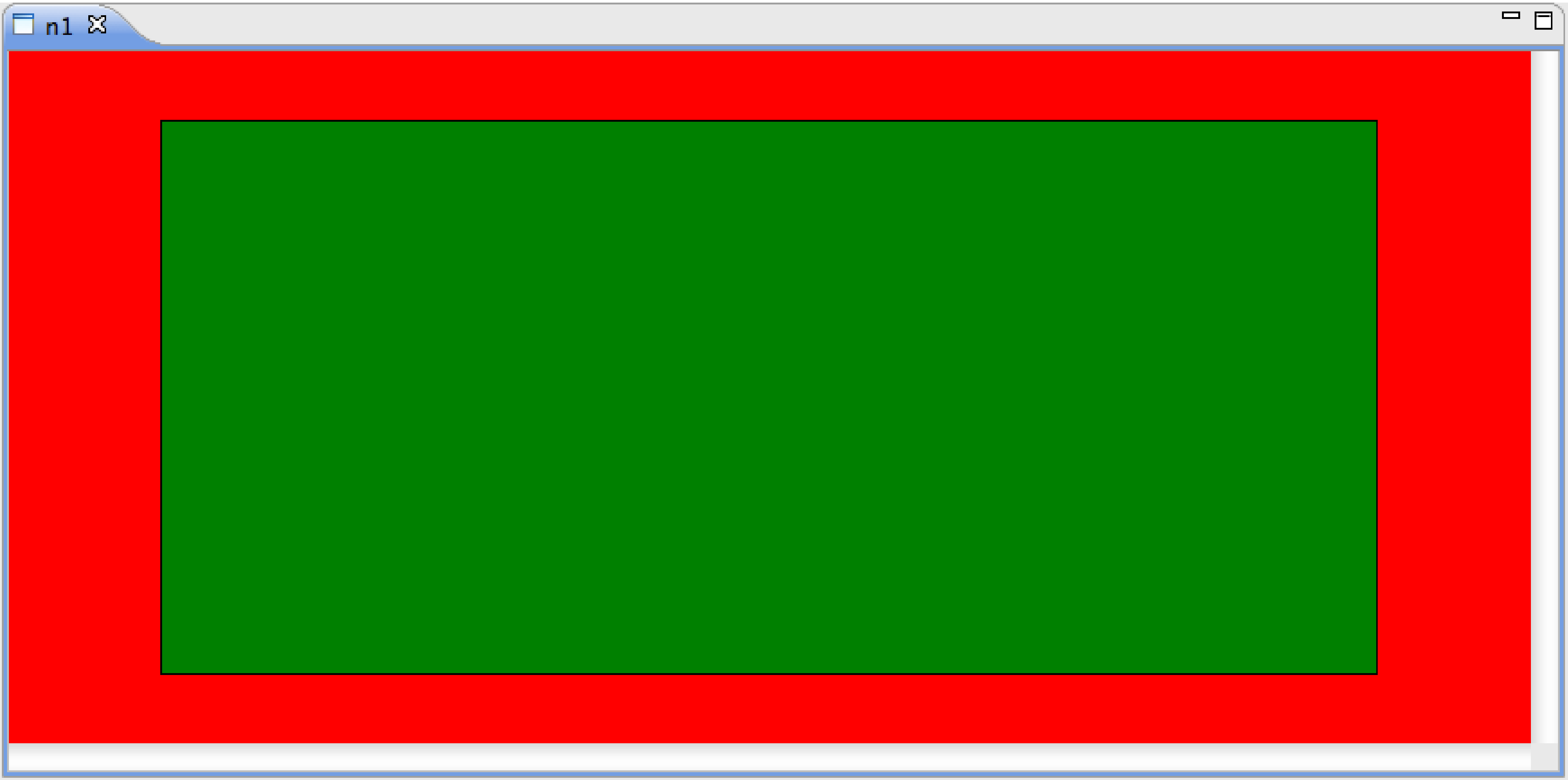
Draw a green rectangle in the right, top corner of the screen that is 20% of the screen size.

Compositional

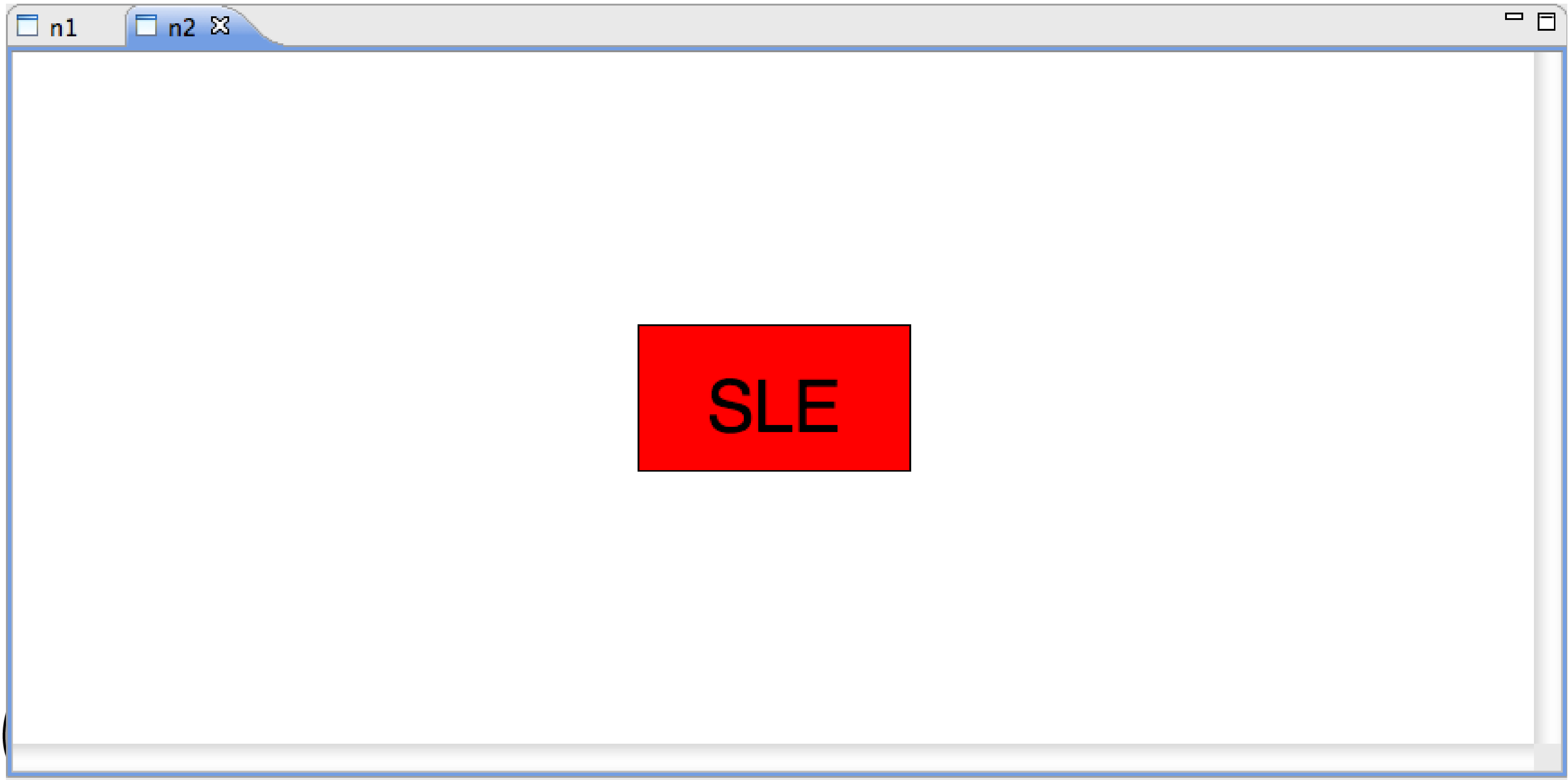
Nesting

- Containers like box and ellipse may contain nested figures
- `shrink`: shrinks figure relative available space of container
- `grow`: enlarges figure relative to size of children
- `resizable`: allow/disallow resizing

```
box(box(fillColor("green"), shrink(0.8)),
    fillColor("red"))
```



```
box(text("SLE", fontSize(40)),
    grow(2), fillColor("red"), resizable(false))
```

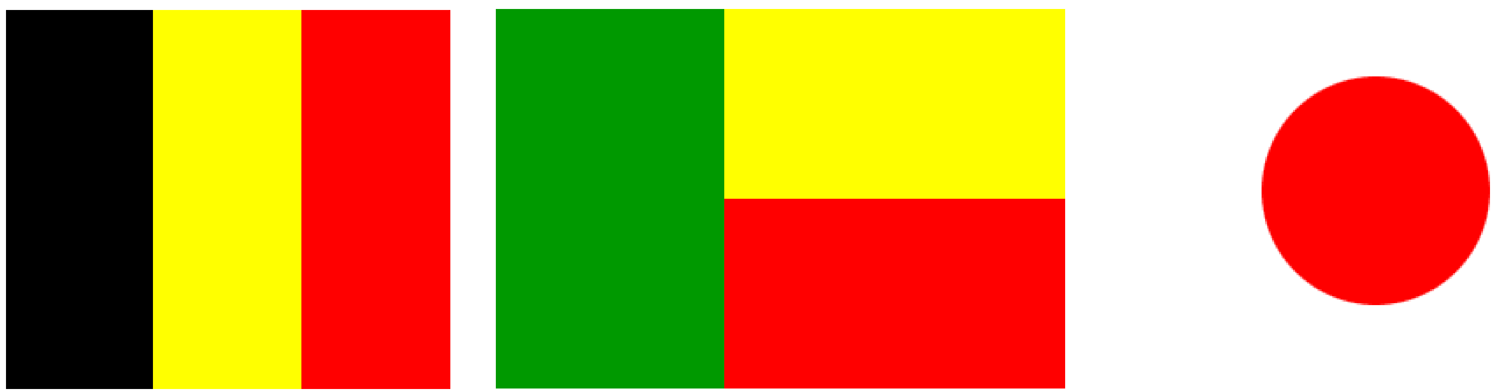


Vertical composition: Dutch Flag

```
B1 = box(fillColor("red"));
B2 = box(fillColor("white"));
B3 = box(fillColor("blue"));
render(vcat([B1, B2, B3]));
```

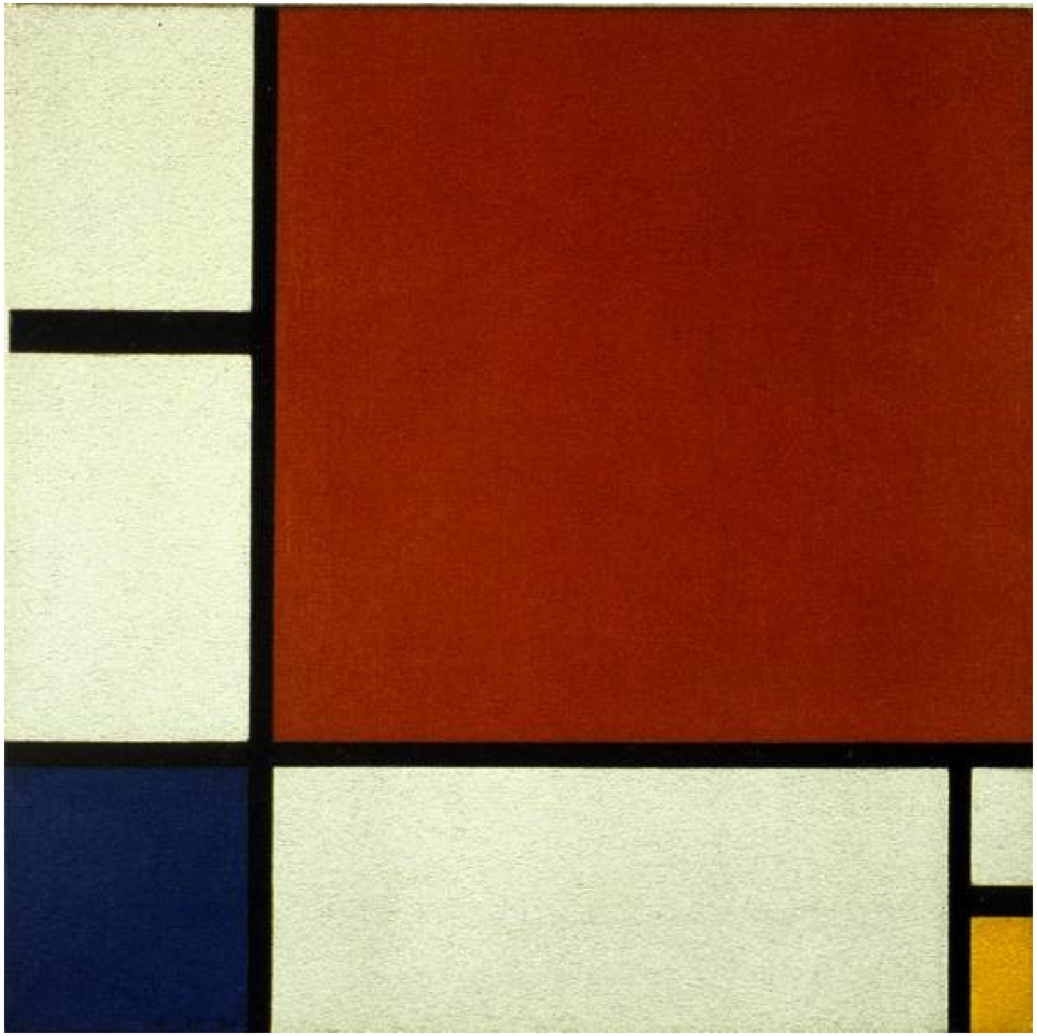
Exercises

- Draw the flags of Belgium, Benin and Japan:



- Write a function `Figure nestedBoxes(int n, real shrink)` that returns `n` nested boxes, each `shrink` smaller than the surrounding one.
  - Hint: use `arbColor()` for random colors.

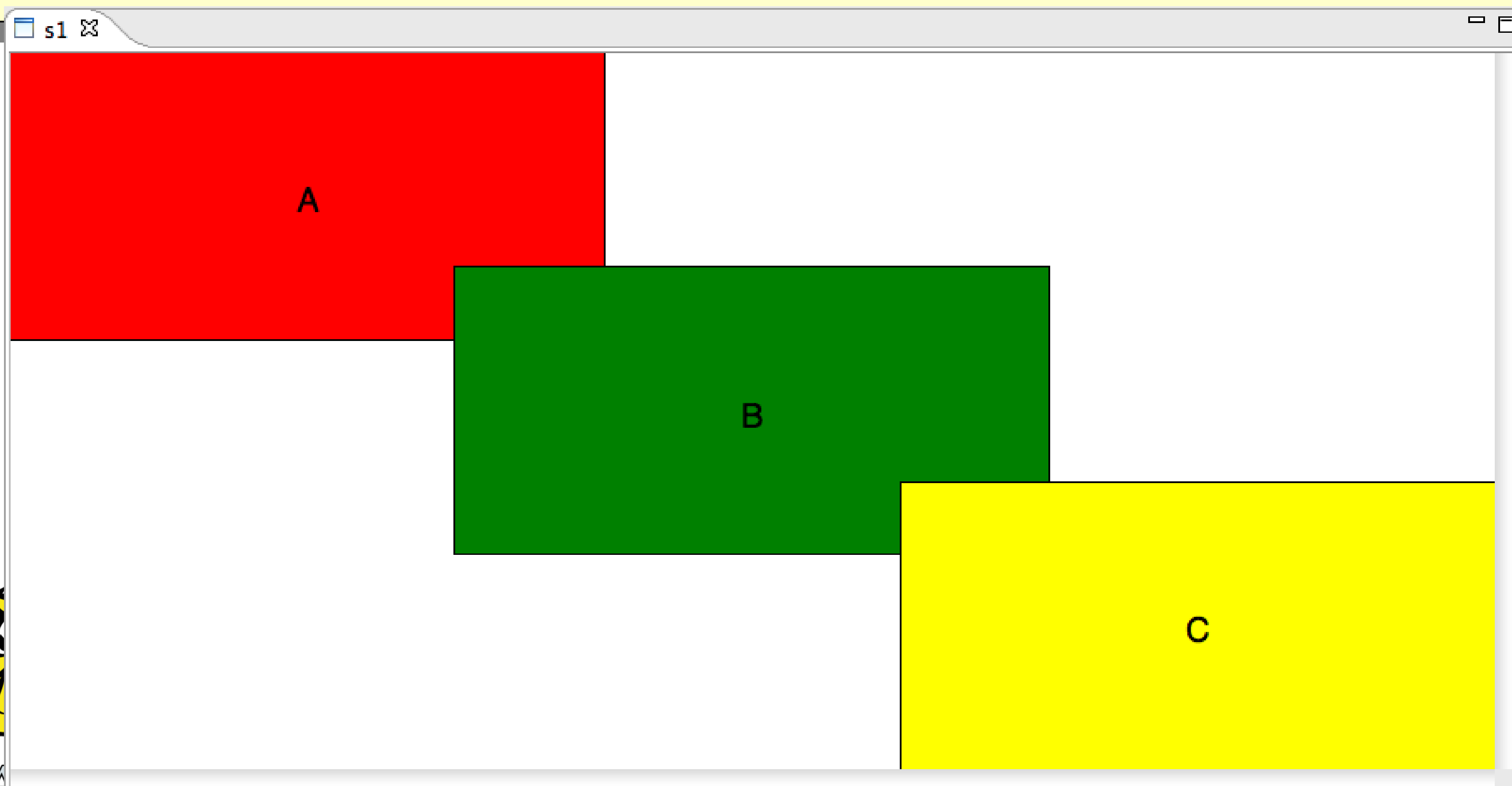
Piet Mondriaan: Composition II in Red, Blue, and Yellow, 1930



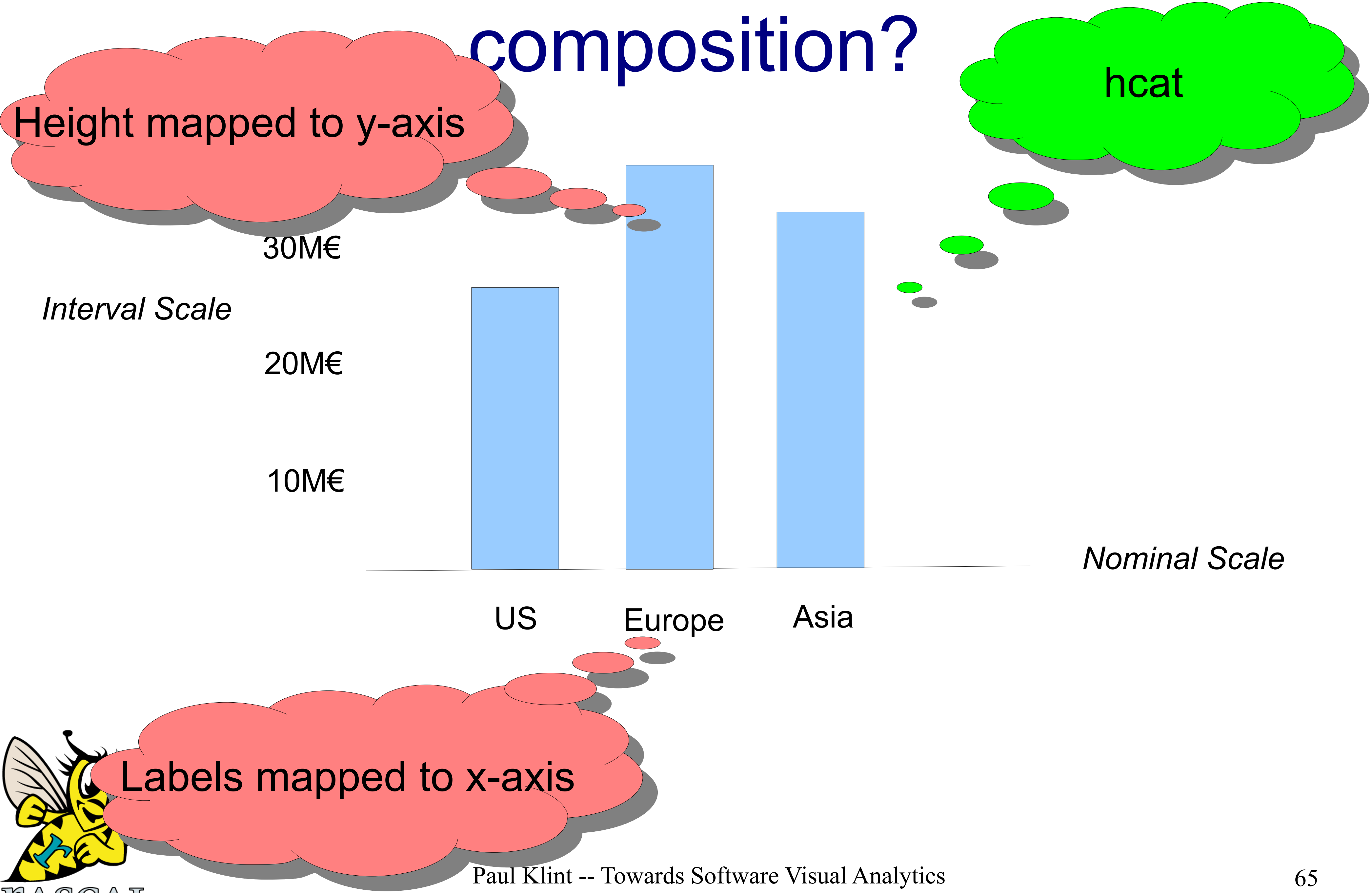
Our simulation (see paper)

Overlay

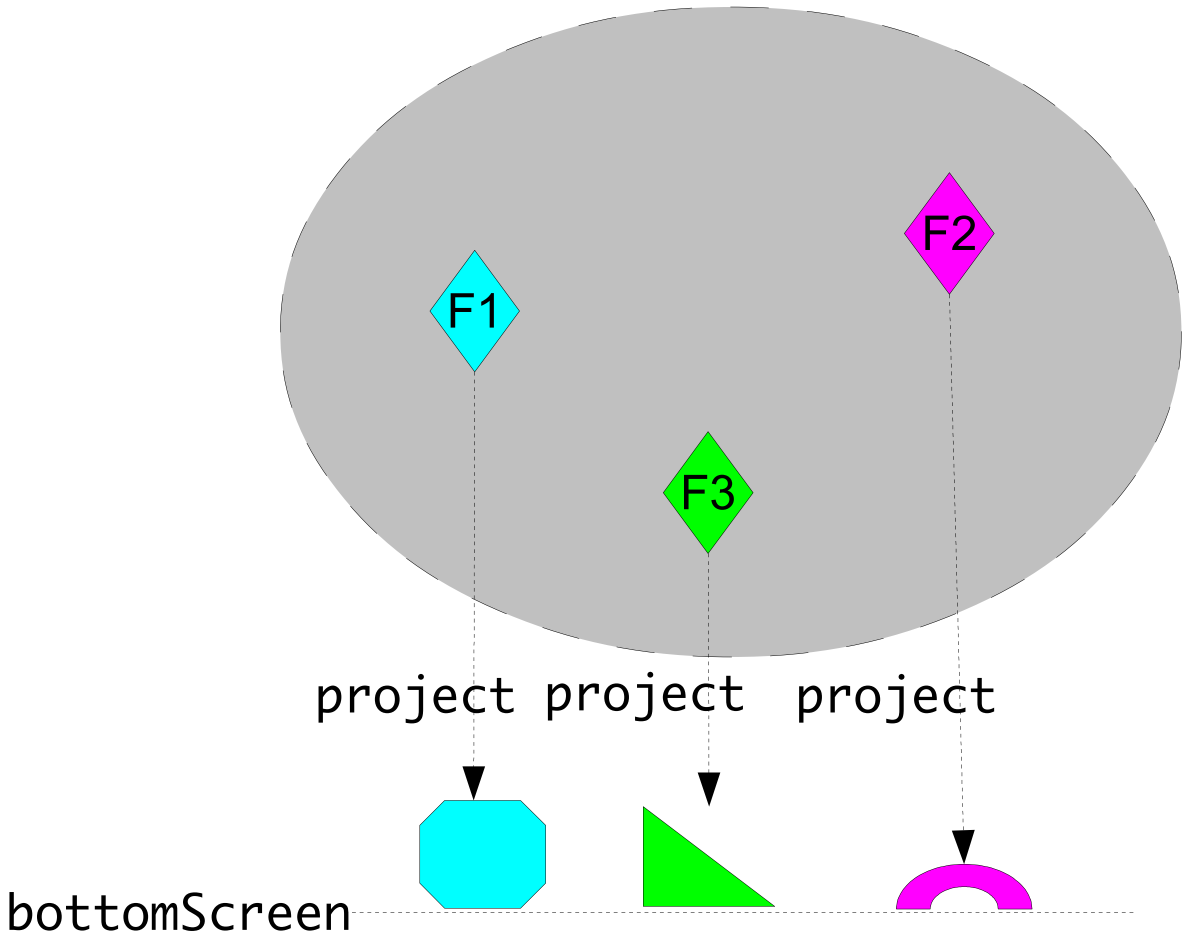
```
overlay([box(text("A"),left(),top(),fillColor("red"),shrink(0.4)),
         box(text("B"),center(),fillColor("green"),shrink(0.4)),
         box(text("C"),right(),bottom(),fillColor("yellow"),
             shrink(0.4))
        ], std(fontSize(20)))
```



How about non-hierarchical composition?



Mapping Figures to Nominal Scale



Arbitrary, nested, Composition of F1, F2 and F3

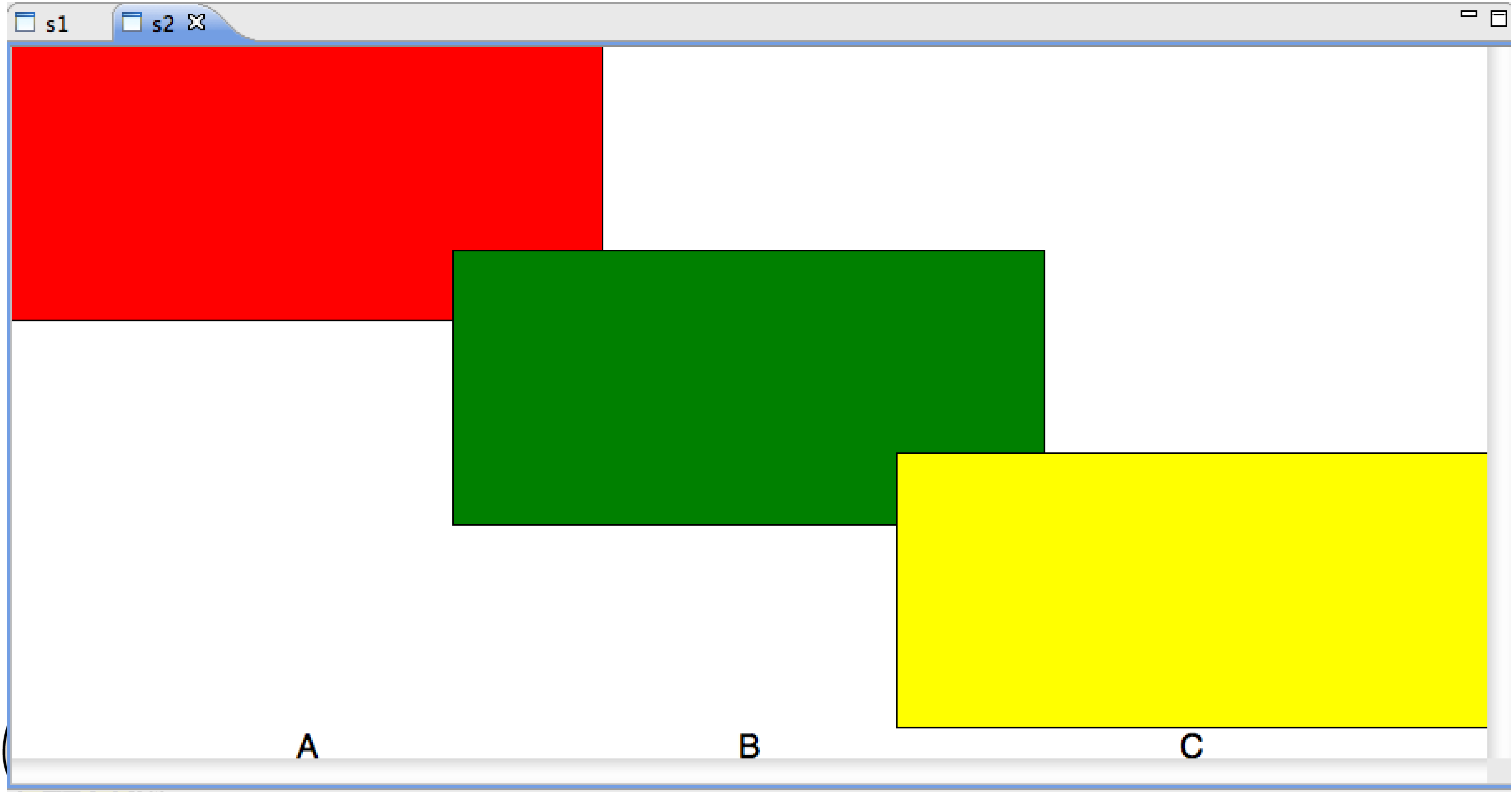
Also available: `topScreen`, `leftScreen`, `rightScreen`

Mapping to a bottomScreen

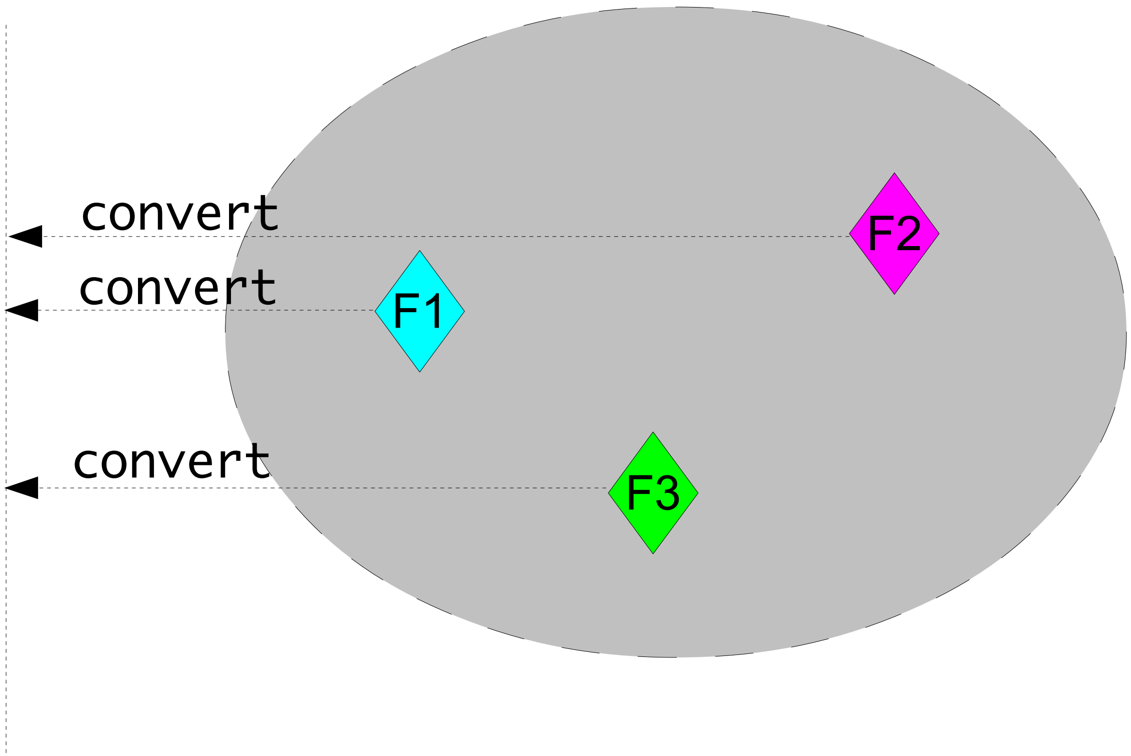
```
overlay(
 [box(        text("A")     ,left(),top(),fillColor("red"),shrink(0.4)),
  box(        text("B")     ,center(),fillColor("green"),shrink(0.4)),
  box(        text("C")     ,right(),bottom(),fillColor("yellow"),shrink(0.4))
 ], stdFontSize(20))
```

```
bottomScreen("s",
 overlay(
  [box(project(text("A"),"s"),left(),top(),fillColor("red"),shrink(0.4)),
   box(project(text("B"),"s"),center(),fillColor("green"),shrink(0.4)),
   box(project(text("C"),"s"),right(),bottom(),fillColor("yellow"),shrink(0.4))
  ], stdFontSize(20))
)
```

Mapping to a bottomScreen



Mapping Figures to Interval Scale



Arbitrary, nested, Composition of F1, F2 and F3

`leftAxis` Map a numeric value of a figure property to an exis

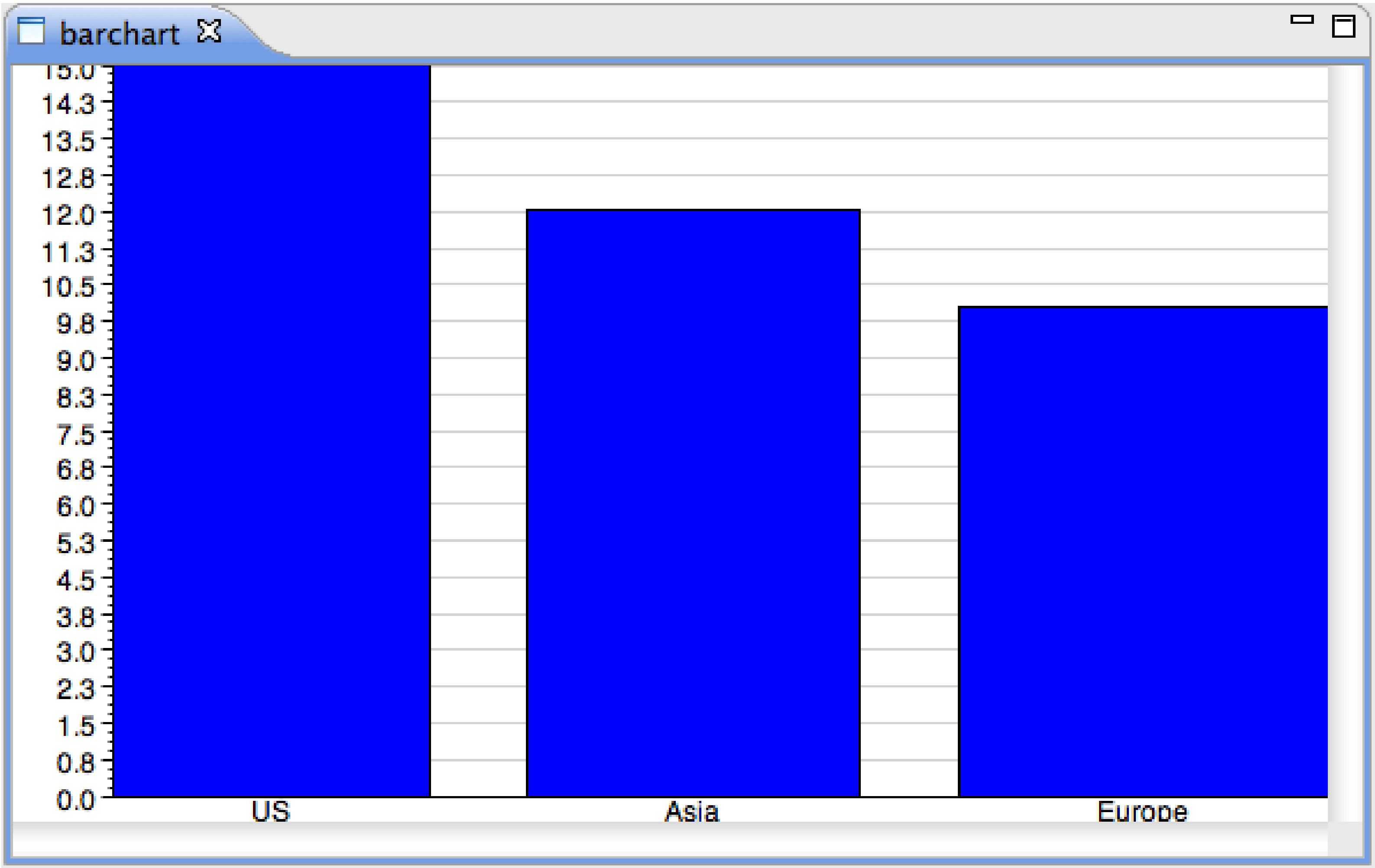
Also available: `rightAxis`, `topAxis`, `bottomAxis`

Barcharts

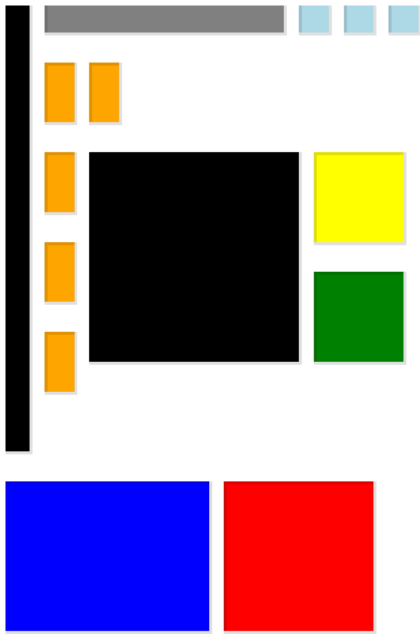
```
public Figure hBarChart(map[str,num] vals){
  return
    bottomScreen("categories",
      leftAxis("y",
                 hcat( [box(height(convert(vals[k],"y")),
                            project(text(k),"categories"),
                            fillColor("blue"))
                       | k <- vals],
                      hgrow(1.2))
    ));
}
```

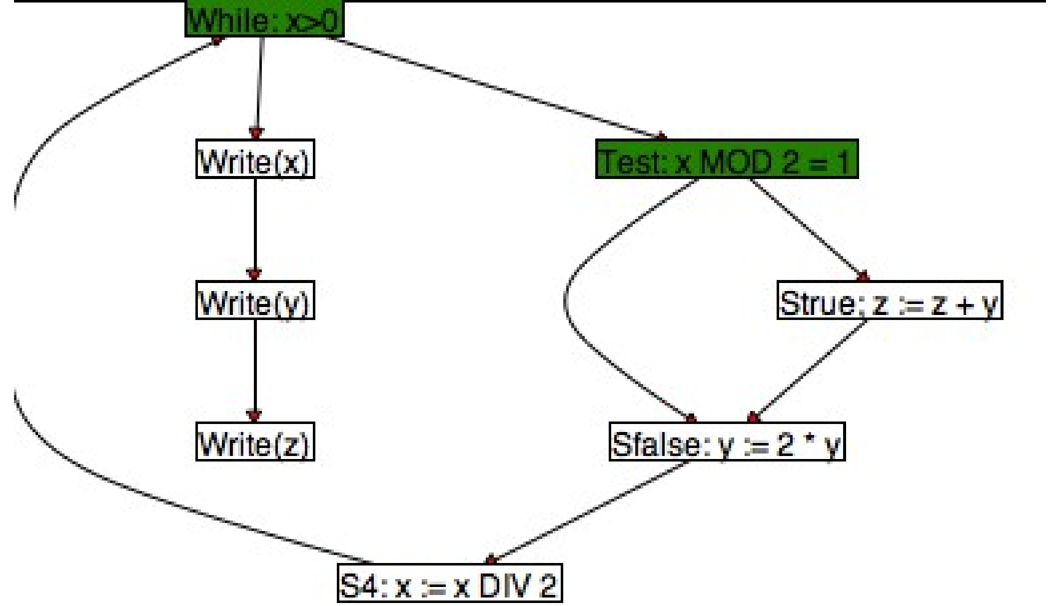
List comprehension that generates a box for each value in the map

```
hBarChart(("Europe" : 10, "US" : 15, "Asia" : 12))
```

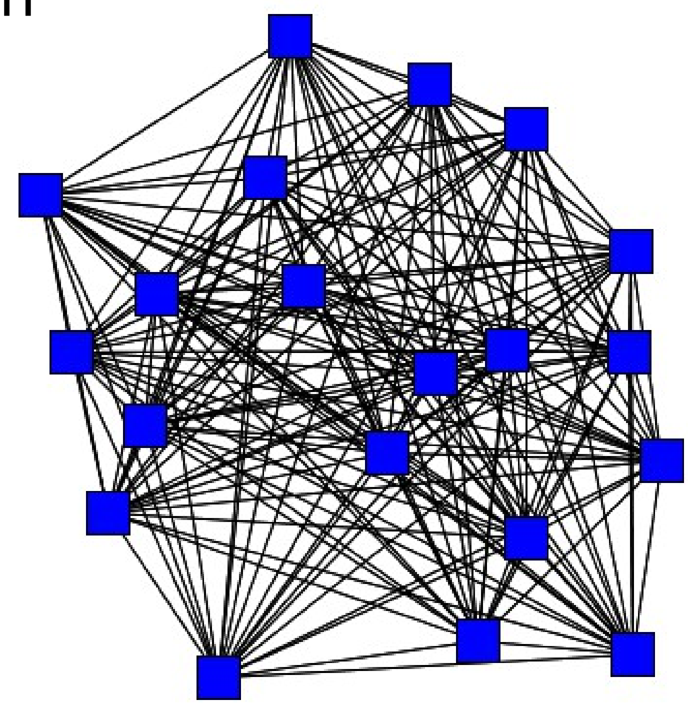


pack

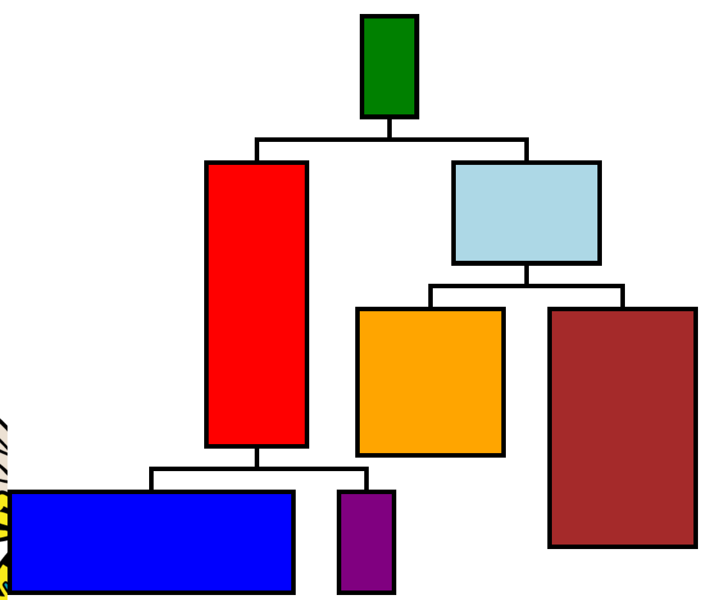




graph



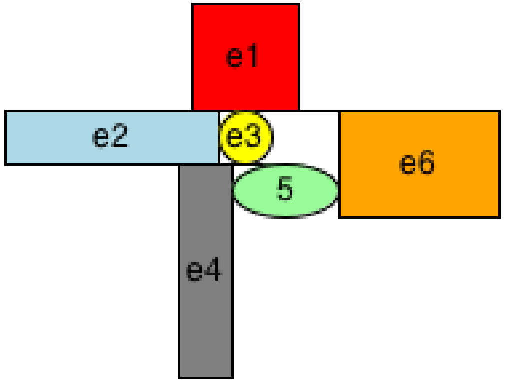
tree

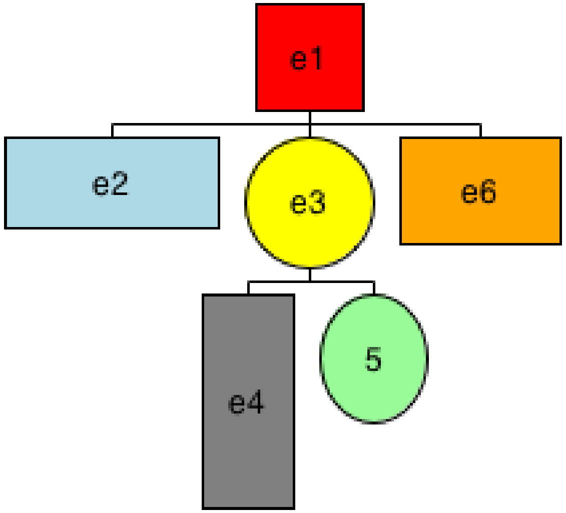


Drawing a Tree

```
tree(e1,
     [ e2,
       tree(e3, [e4, e5]),
       e6
     ]
    );
```

```
tree(e1,
     [ e2,
       tree(e3, [e4, e5]),
       e6
     ], std(gap(10))
    );
```





Example: World Energy Ltd.

- Given is the following company information:
  - Company name
  - Divisions:
    - Division name
    - List of Units:
      - Unit name
      - Number of employees in unit
      - Profit made by unit
- Challenge: show company structure and emphasize profit per employee

Representing a Company

```
data COMPANY = company(str name, list[DIVISION] divisions);

data DIVISION = division(str name, list[UNIT] units);

data UNIT = unit(str name, int employees, int profit);
```

```
public COMPANY we =
   company("World Energy Ltd",
   [ division("Traditional",
              [ unit("Oil", 1000, 20000000),
                unit("Gas", 2000, 15000000)
              ]),
     division("Eco",
              [ unit("Wind", 500, 1000000),
                unit("Sun",  300, 3000000),
                unit("Bio",  100, 1050000)
              ]),
     division("Research",
              [ unit("Hydro", 50, 450000),
                unit("Earth", 80, 200000)
              ])
   ] );
```

Fact Extraction

```
public tuple[int, int] totals(COMPANY c){
  nemp = 0;
  nprof = 0;
  for(/unit(name, emp, prof) <- c){
     nemp += emp;
     nprof += prof;
  }
  return <nemp, nprof>;
}
```

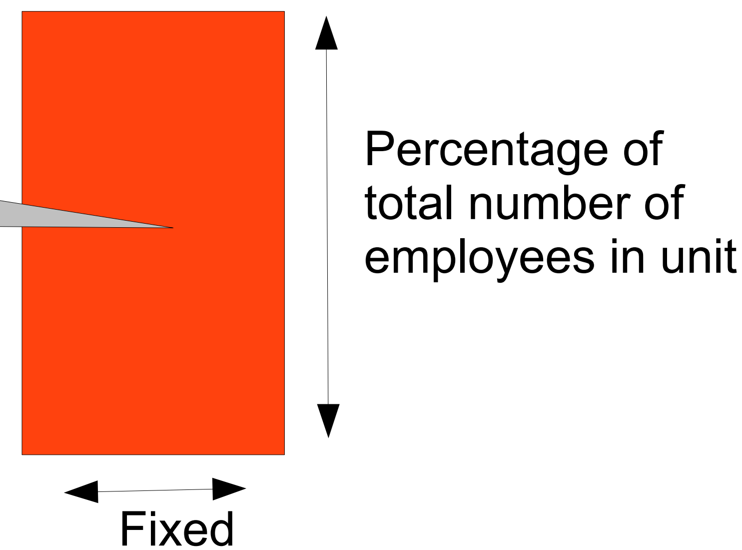
```
totals(we) => <4030,38000000>
```

Visualize a Unit

```
Figure drawUnit(UNIT u, int totalEmployees, int totalProfit){
    percEmp = 100 * u.employees / totalEmployees;  // percentage employees in unit
    avg = 100 * totalProfit / totalEmployees;
    thisAvg = 100 * u.profit / u.employees;         // average profit per employee in unit
    c = mapColor1(avg, thisAvg);
    return box(text(u.name), size(50, 4*percEmp), resizable(false), fillColor(c));
}
```

```
Color mapColor1(num avg, num uAvg) =
   uAvg < avg ? color("red") : color("green");
```

Color coding
Red: profit/employee below average
Green: profit/employee average average

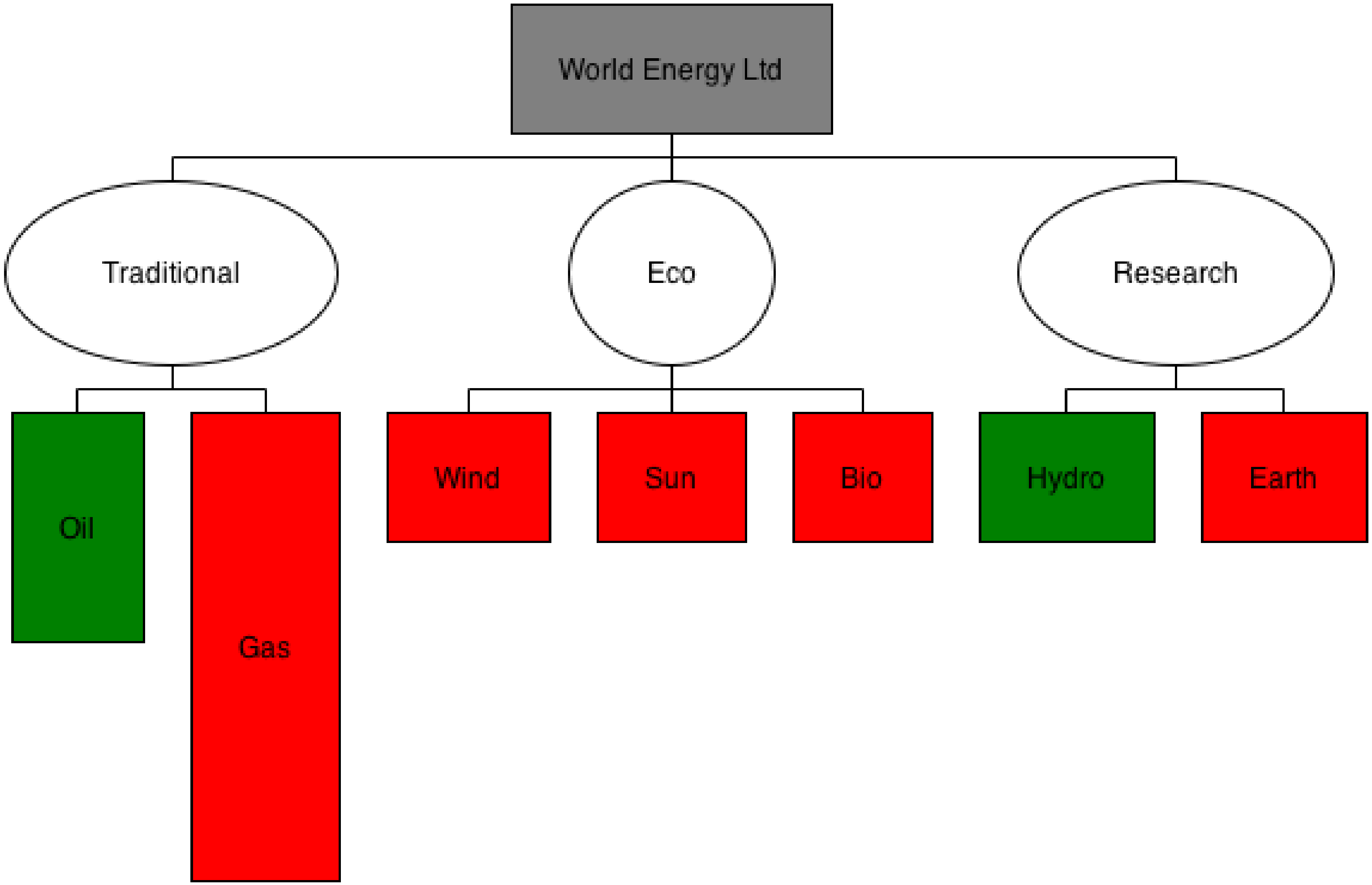


Visualize a Company

```
public Figure drawCompany(COMPANY c){
  <nemp, nprof> = totals(c);
  return tree(box(text(c.name), fillColor("grey")),
              [drawDivision(d, nemp, nprof) | d <- c.divisions],
              std(gap(20)));
}

Figure drawDivision(DIVISION div, int totalEmployees, int totalProfit){
    return tree(ellipse(text(div.name)),
               [drawUnit(u, totalEmployees, totalProfit) | u <- div.units]);
}
```

render(we)



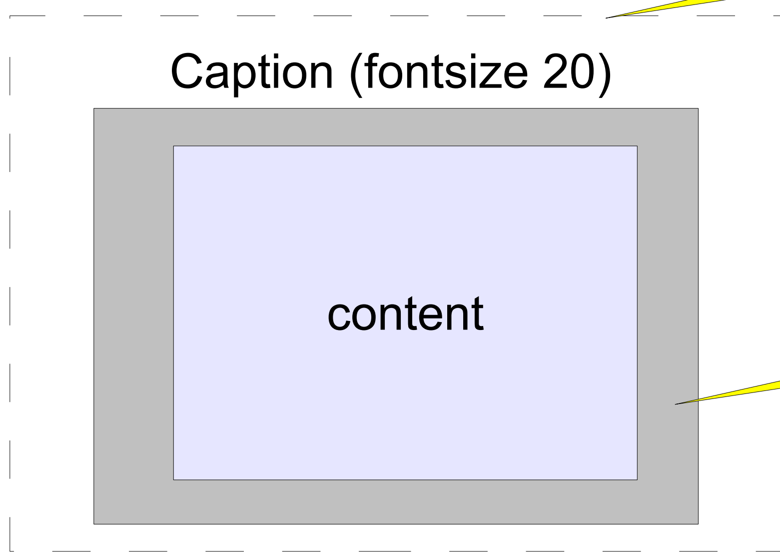
World Energy Ltd
Traditional
Eco
Research
Oil
Gas
Wind
Sun
Bio
Hydro
Earth

Caption and Frame

Draw shadows

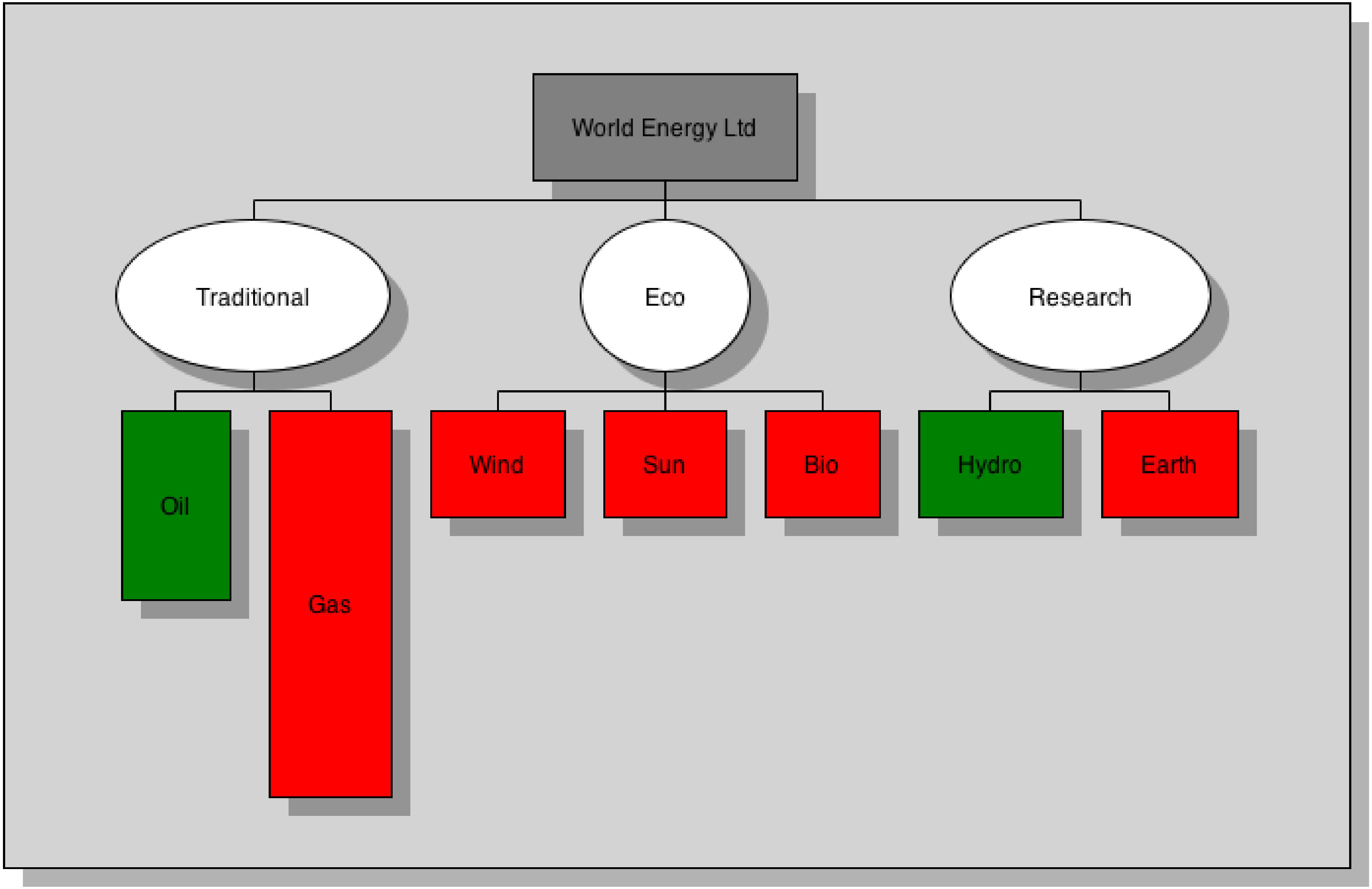
```
public Figure frame(str caption, Figure content){
  return vcat([ text(caption, fontSize(20)),
                box(content, grow(1.2), std(shadow(true)), fillColor("lightgrey"))
              ], shrink(0.5));
}
```

0.5 x smaller than screen



1.2 x larger than content

Profit/Employee mapped on company divisions



Interactive

How to add user interaction?

- In a query/exploration setting the user wants control over parts of the visualization:
  - Get feedback when mouse hovers over a figure
  - Enter search strings
  - Enable/disable detailed views
  - Zoom in to get details
  - Etc.
- How can user interaction be fitted into the current model?

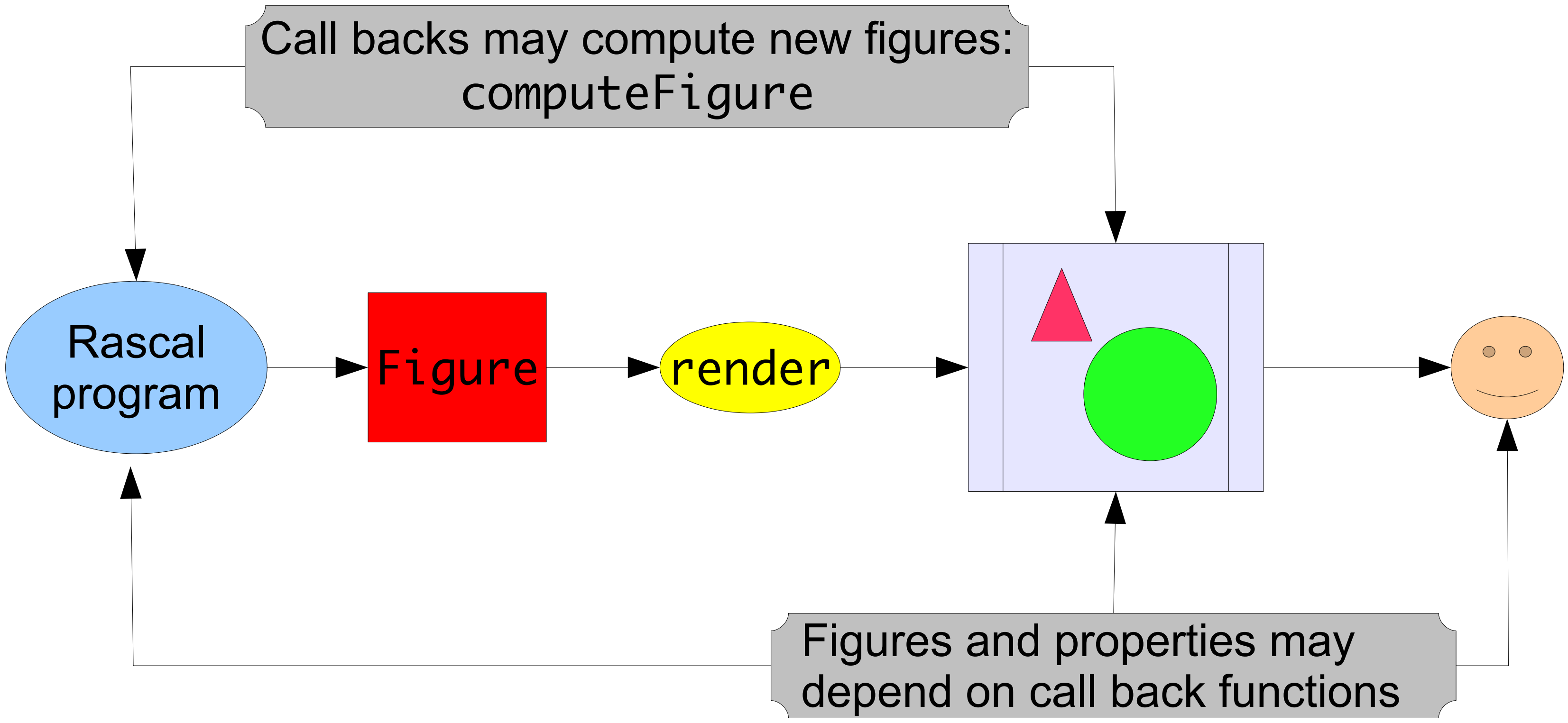
Approach

- Not a complete GUI construction kit, but our composition primitives easily match what “layout managers” in GUI toolkits do
- All application data stay in the Rascal program
- Properties can have “computed” values:
  - `width(int w)`
  - `width(int() w)`

w is int function without arguments thats computes width when needed

- Add interaction primitives that use Rascal functions as call back

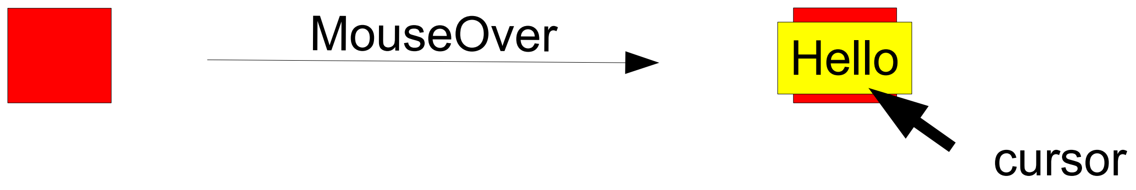
Interaction Architecture



mouseOver

```
public FProperty popup(str S){
  return mouseOver(box(text(S), fillColor("lightyellow"),
                       grow(1.2),resizable(false)));
}
```

```
box(size(50),fillColor("red"), popup("Hello"))
```



MouseOvers can be arbitrarily nested

textfield

Initial text of field

Function to call when
Text entry is complete

```
Figure textfield(str text,  void (str) callback,
                            bool (str) validate, FProperty props...)
```

Optional function function to validate
(partially) entered text

Optional properties

Example: box with user-controlled height

A red box

```
box(width(100), height(100), fillColor("red"))
```

A red box, height controlled by variable H

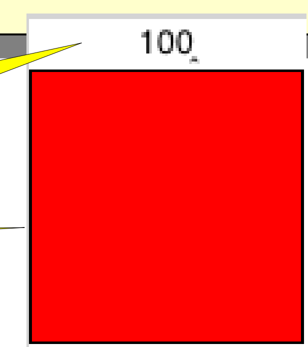
```
box(width(100), height(int() { return H; }), fillColor("red"))
```

A red box, with input field to control H

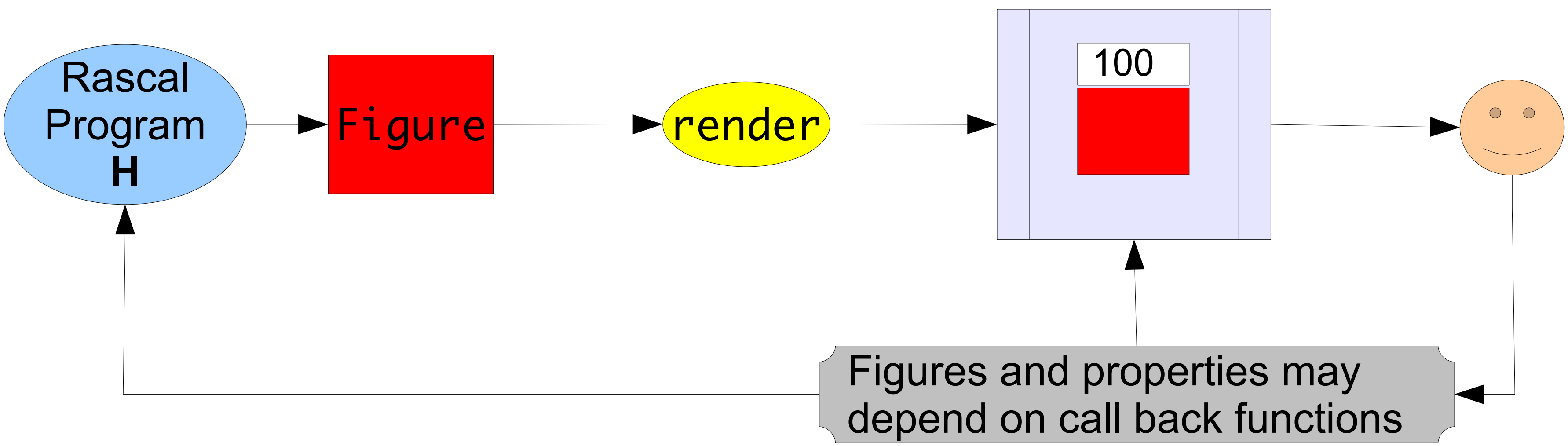
```
int H = 100;
fig = vcat([ textfield("<H>", void(str s){H = toInt(s);}, intInput, size(100,15)),
             box(width(100),height(int(){return H;}),fillColor("red"))
           ]);
```

Editable Text field

Height changes With value of text field



Example:
box with user-controlled height



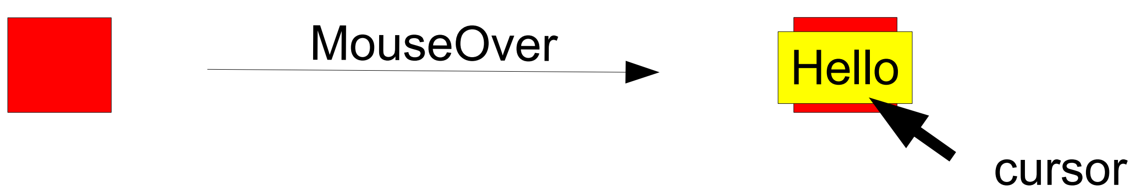
More interaction

- `button`: handle button press
- `onClick`: handle mouse clicks on this figure
- `computeFigure`: compute new figure on demand
- `selectFigure`: select from precomputed figures
- `choice`: from alternatives (drop down menu)

FProperty: mouseOver

```
public FProperty popup(str s){
  return mouseOver(box(text(s), gap(1), fillColor("yellow")));
}
```

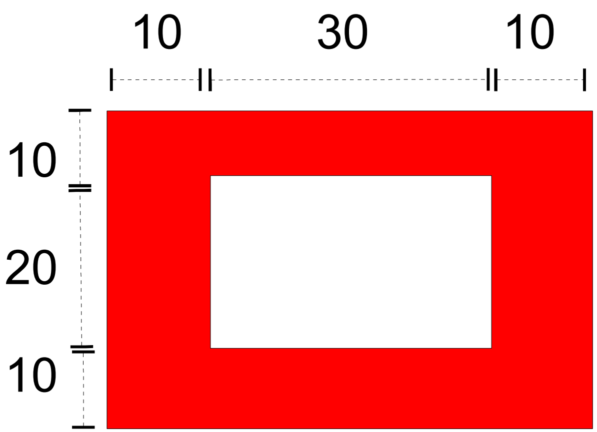
```
box(size(10), color("red"), popup("Hello"))
```



Size and Nesting

Inner Box

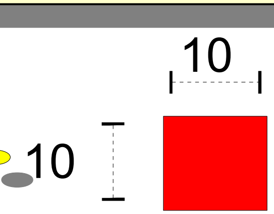
```
box(box(size(30, 20), color("white")), gap(10), color("red"))
```



Size of outer box determined by size of inner box

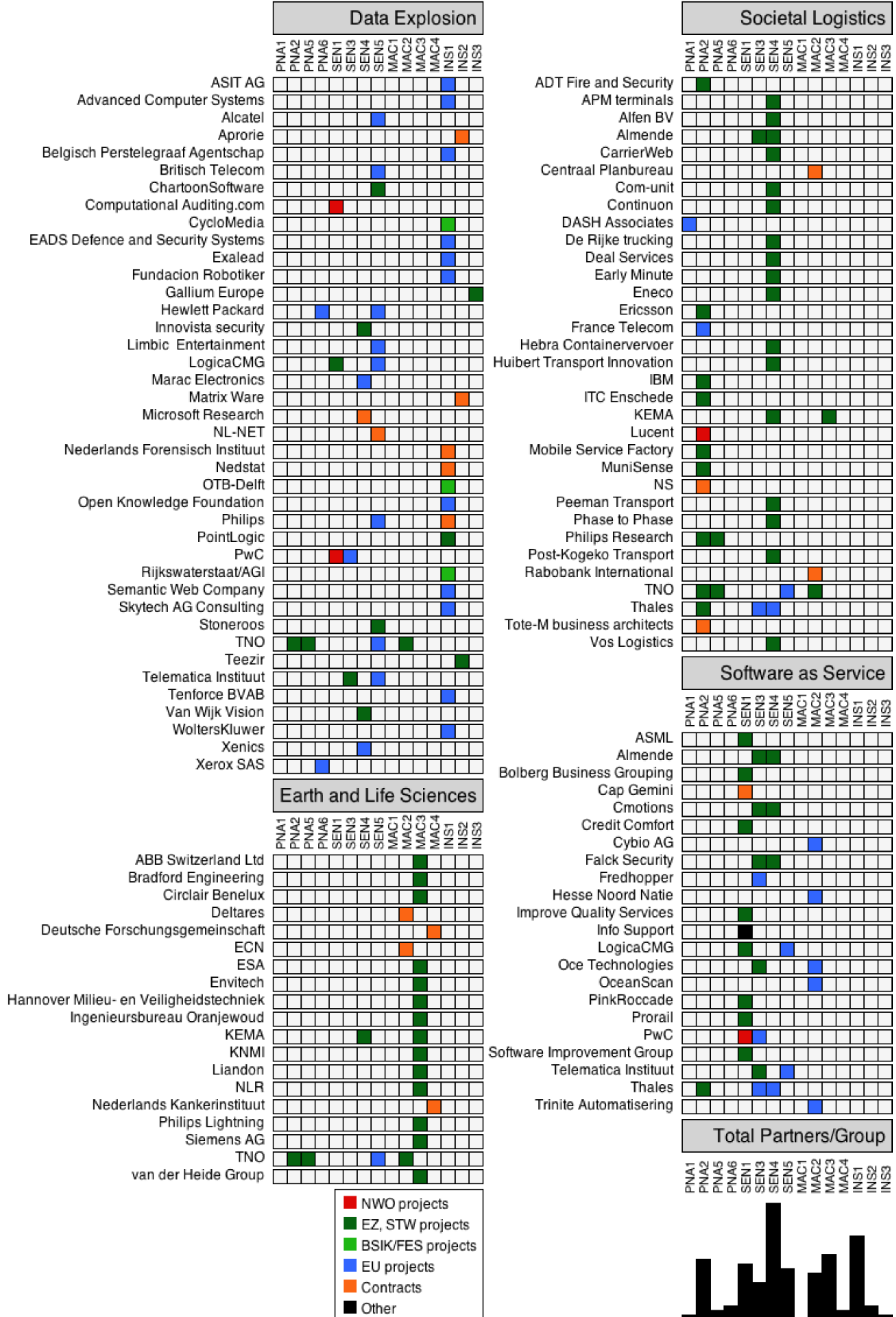
```
box(box(size(30, 20)), size(10), gap(10), color("red"))
```

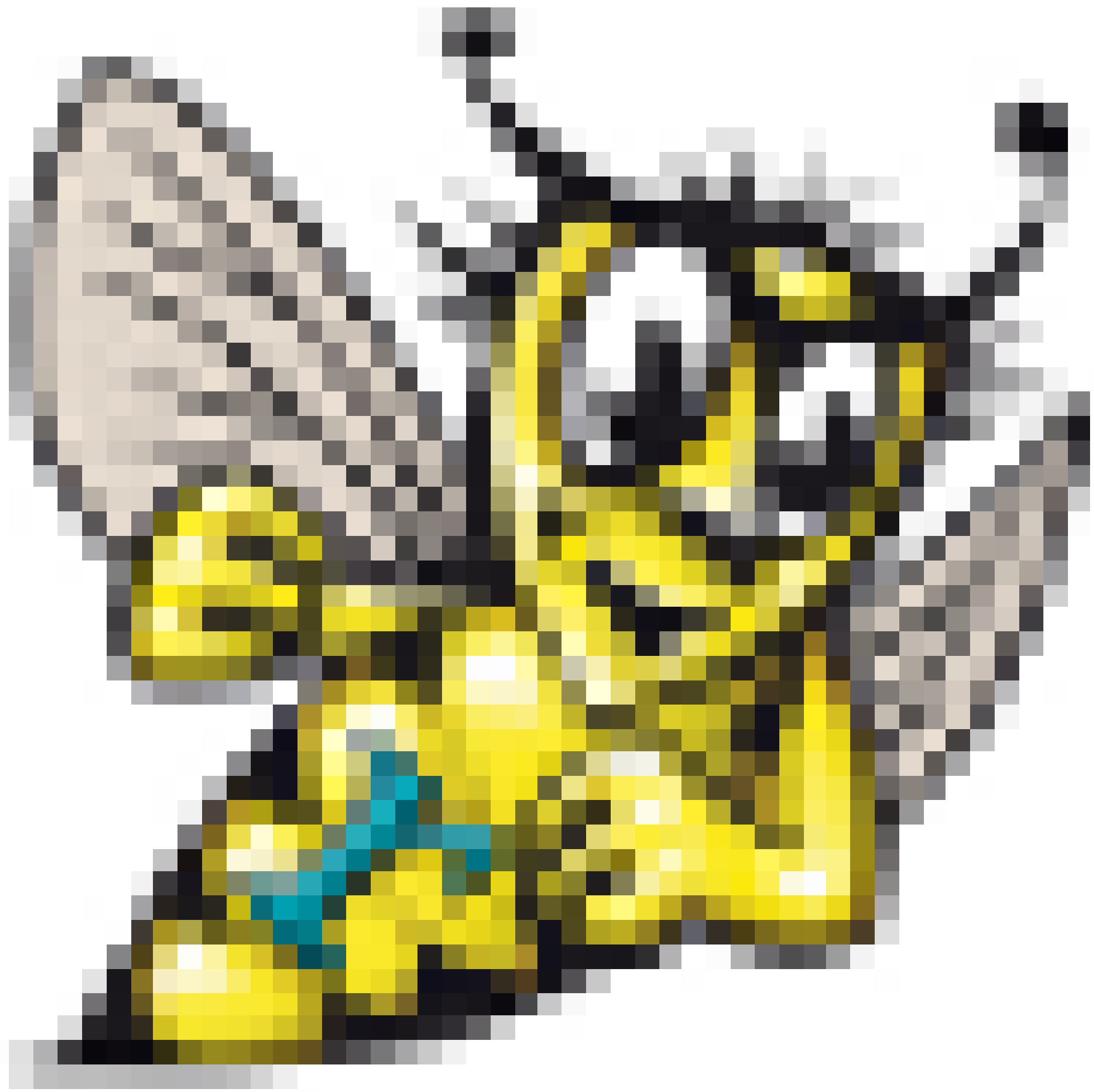
Inner box does not fit

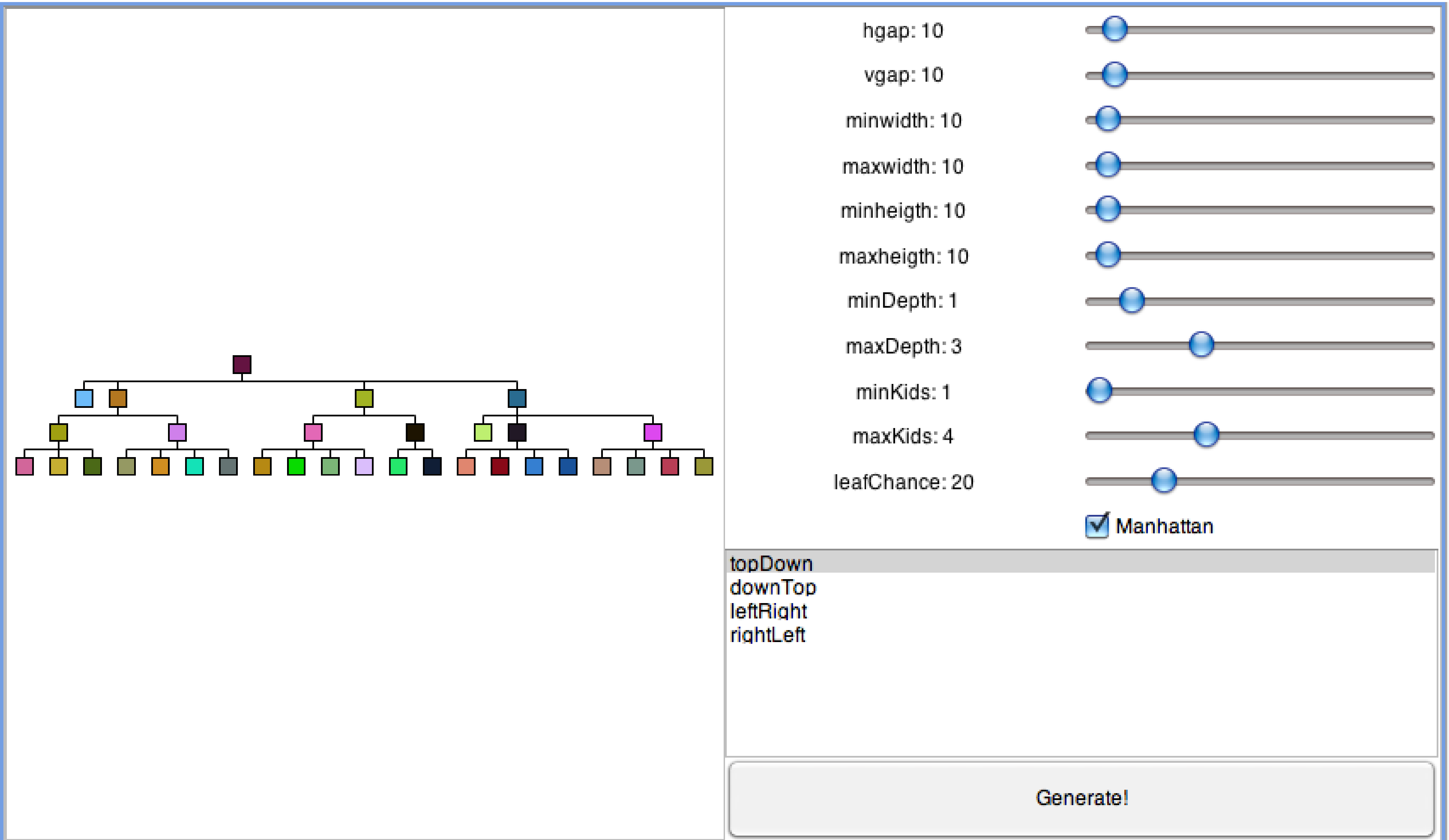


... but will appear on MouseOver

Examples

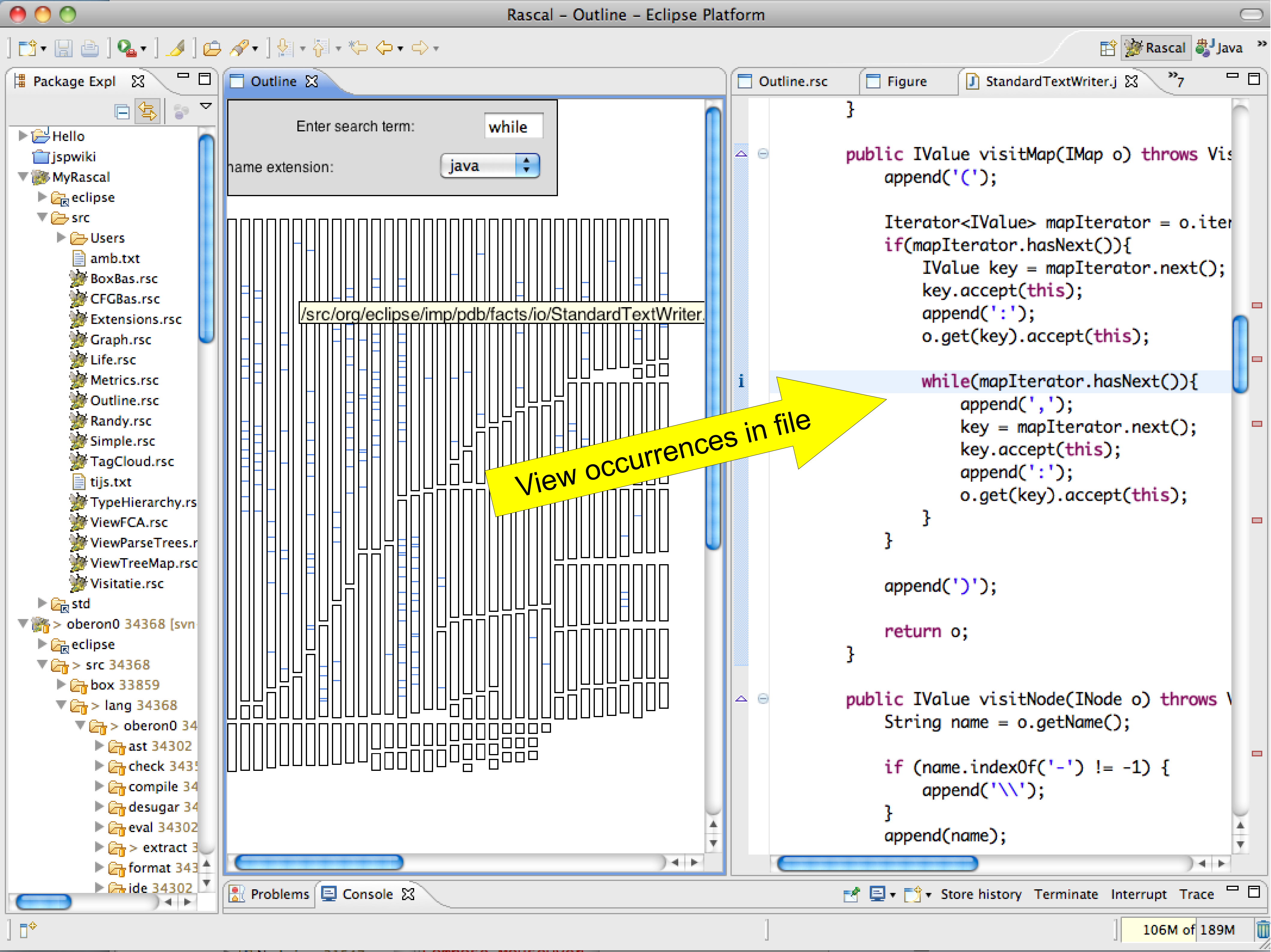

Data Explosion
PNA1 PNA2 PNA5 PNA6 SEN1 SEN3 SEN4 SEN5 MAC1 MAC2 MAC3 MAC4 INS1 INS2 INS3
ASIT AG
Advanced Computer Systems
Alcatel
Aprorie
Belgisch Perstelegraaf Agentschap
Britisch Telecom
ChartoonSoftware
Computational Auditing.com
CycloMedia
EADS Defence and Security Systems
Exalead
Fundacion Robotiker
Gallium Europe
Hewlett Packard
Innovista security
Limbic Entertainment
LogicaCMG
Marac Electronics
Matrix Ware
Microsoft Research
NL-NET
Nederlands Forensisch Instituut
Nedstat
OTB-Delft
Open Knowledge Foundation
Philips
PointLogic
PwC
Rijkswaterstaat/AGI
Semantic Web Company
Skytech AG Consulting
Stoneroos
TNO
Teezir
Telematica Instituut
Tenforce BVAB
Van Wijk Vision
WoltersKluwer
Xenics
Xerox SAS
Earth and Life Sciences
PNA1 PNA2 PNA5 PNA6 SEN1 SEN3 SEN4 SEN5 MAC1 MAC2 MAC3 MAC4 INS1 INS2 INS3
ABB Switzerland Ltd
Bradford Engineering
Circlair Benelux
Deltares
Deutsche Forschungsgemeinschaft
ECN
ESA
Envitech
Hannover Milieu- en Veiligheidstechniek
Ingenieursbureau Oranjewoud
KEMA
KNMI
Liandon
NLR
Nederlands Kankerinstituut
Philips Lightning
Siemens AG
TNO
van der Heide Group
NWO projects
EZ, STW projects
BSIK/FES projects
EU projects
Contracts
Other
Societal Logistics
PNA1 PNA2 PNA5 PNA6 SEN1 SEN3 SEN4 SEN5 MAC1 MAC2 MAC3 MAC4 INS1 INS2 INS3
ADT Fire and Security
APM terminals
Alfen BV
Almende
CarrierWeb
Centraal Planbureau
Com-unit
Continuon
DASH Associates
De Rijke trucking
Deal Services
Early Minute
Eneco
Ericsson
France Telecom
Hebra Containervervoer
Huibert Transport Innovation
IBM
ITC Enschede
KEMA
Lucent
Mobile Service Factory
MuniSense
NS
Peeman Transport
Phase to Phase
Philips Research
Post-Kogeko Transport
Rabobank International
TNO
Thales
Tote-M business architects
Vos Logistics
Software as Service
PNA1 PNA2 PNA5 PNA6 SEN1 SEN3 SEN4 SEN5 MAC1 MAC2 MAC3 MAC4 INS1 INS2 INS3
ASML
Almende
Bolberg Business Grouping
Cap Gemini
Cmotions
Credit Comfort
Cybio AG
Falck Security
Fredhopper
Hesse Noord Natie
Improve Quality Services
Info Support
LogicaCMG
Oce Technologies
OceanScan
PinkRoccade
Prorail
PwC
Software Improvement Group
Telematica Instituut
Thales
Trinite Automatisering
Total Partners/Group
PNA1 PNA2 PNA5 PNA6 SEN1 SEN3 SEN4 SEN5 MAC1 MAC2 MAC3 MAC4 INS1 INS2 INS3

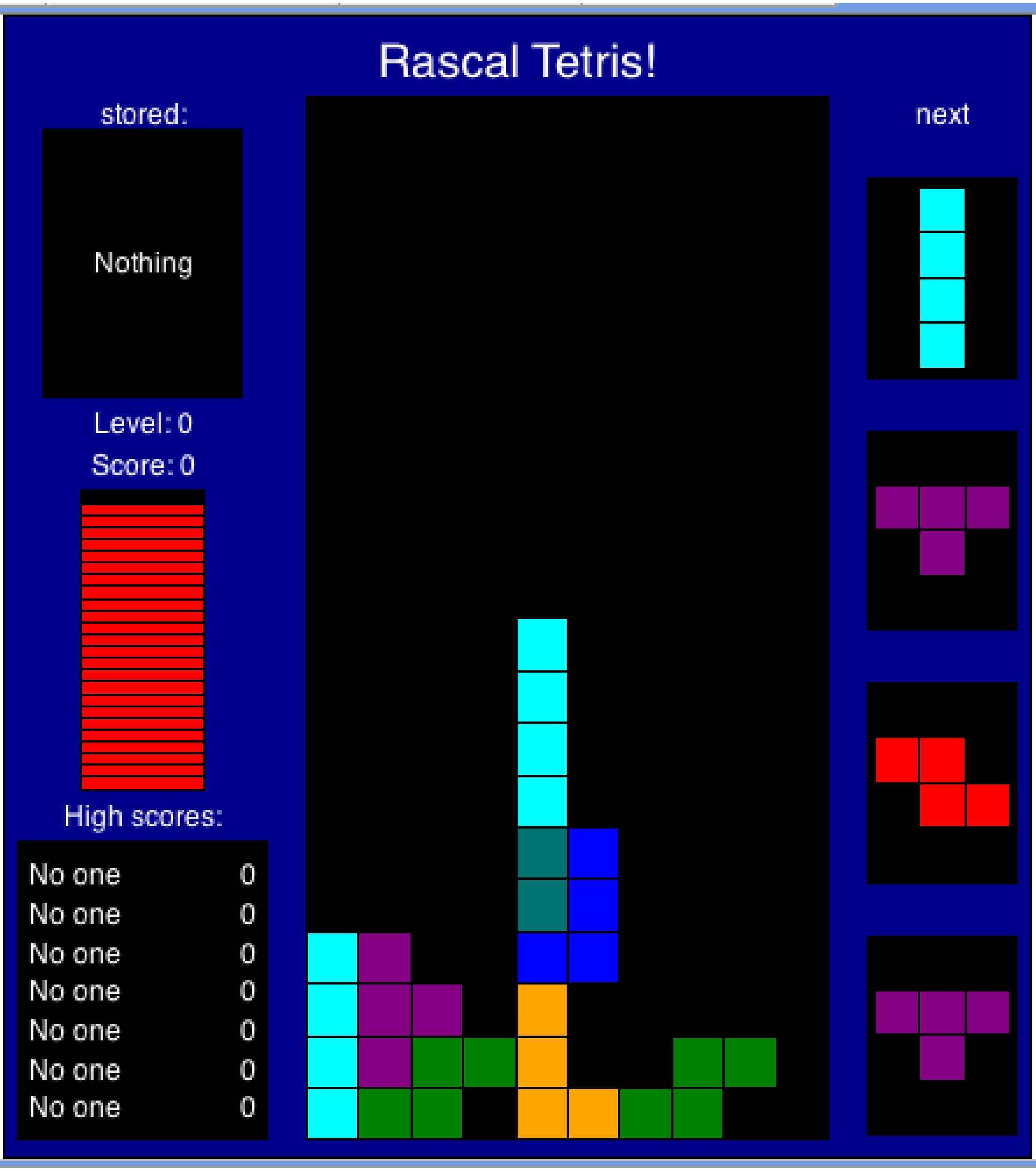



hgap: 10
vgap: 10
minwidth: 10
maxwidth: 10
minheigth: 10
maxheigth: 10
minDepth: 1
maxDepth: 3
minKids: 1
maxKids: 4
leafChance: 20
Manhattan
topDown
downTop
leftRight
rightLeft
Generate!

While working on a Java project ...

- What are the different file types used in this project?
- How are the following properties distributed:
  - Number of attributes/class
  - Number of methods/class
  - Number of implemented interfaces/class


Rascal – Outline – Eclipse Platform
Rascal
Java
Package Expl
Hello
jspwiki
MyRascal
eclipse
src
Users
amb.txt
BoxBas.rsc
CFGBas.rsc
Extensions.rsc
Graph.rsc
Life.rsc
Metrics.rsc
Outline.rsc
Randy.rsc
Simple.rsc
TagCloud.rsc
tijs.txt
TypeHierarchy.rs
ViewFCA.rsc
ViewParseTrees.r
ViewTreeMap.rsc
Visitatie.rsc
std
> oberon0 34368 [svn
eclipse
> src 34368
box 33859
> lang 34368
> oberon0 34
ast 34302
check 343
compile 34
desugar 34
eval 34302
> extract 3
format 343
ide 34302
Outline
Enter search term:
while
name extension:
java
/src/org/eclipse/imp/pdb/facts/io/StandardTextWriter.
Outline.rsc
Figure
StandardTextWriter.j
View occurrences in file
}
public IValue visitMap(IMap o) throws Vis
append('(');
Iterator<IValue> mapIterator = o.iter
if(mapIterator.hasNext()){
IValue key = mapIterator.next();
key.accept(this);
append(':');
o.get(key).accept(this);
while(mapIterator.hasNext()){
append(',');
key = mapIterator.next();
key.accept(this);
append(':');
o.get(key).accept(this);
}
}
append(')');
return o;
}
public IValue visitNode(INode o) throws V
String name = o.getName();
if (name.indexOf('-') != -1) {
append('\\');
}
append(name);
Problems
Console
Store history
Terminate
Interrupt
Trace
106M of 189M


Rascal Tetris!
stored:
Nothing
Level: 0
Score: 0
High scores:
No one 0
No one 0
No one 0
No one 0
No one 0
No one 0
No one 0
next

Summary

- The **vis::Figure** library provides a small core with powerful composition operators
- Already supports many interactive visualization needs
- Integrated in main stream language (Java) and IDE (Eclipse)

Resources

- Stephan Diehl, Software Visualization, Springer, 2007
- Edward Tufte, The Visual Display of Quantitative Information, Graphics Press, 2001
- Jeffrey Heer, Michael Bostock, Vadim Ogievetsky, A tour through the visualization zoo, CACM, 2010 vol. 53 (6)