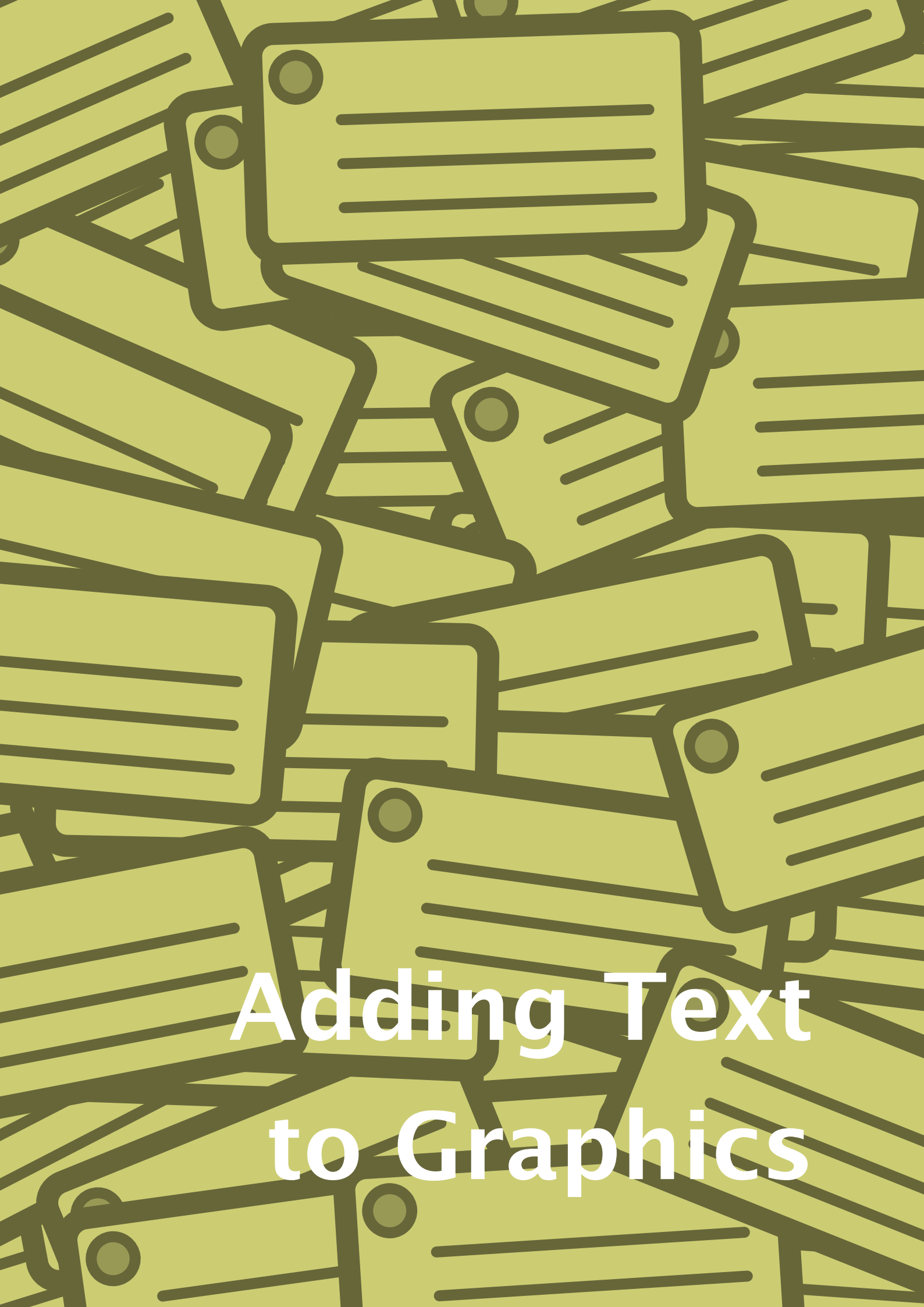


Adding Text to Graphics

Introduction

This is short manual about adding text to graphics made by other applications than TEX. Early versions of CONTEXT already had provisions for adding information to graphics. The `\start...\stopfigure` environment provides a way to add text and hyperlinks to graphics based on a grid. We used this feature to create interactive maps, navigate using diagrams and alike. The corresponding definitions are stored per graphic, and can be managed independently from the main text.

The method described in this document reimplements this feature in a more flexible way using a couple of features not present at that time. Also, the new method is more suited to handle information stored in database like the figure and resource databases supported by CONTEXT. In due time the old mechanism will be replaced by (i.e. redefined in) the new one.

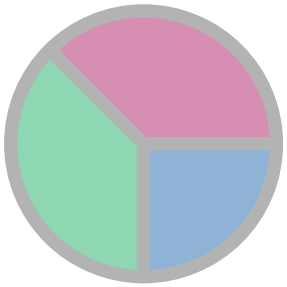
The reason for extending the figure database concept with this kind of information is that the people responsible for the content (text) are not always the same as those making the graphics. In some of our projects, authors are supposed to add text (here called labels) to graphics. The same graphic can be used in more than one context, with different labels. Think for instance of a graphic that is used in a question without labels, but in an answer with labels. Or consider the same graphic being used in a Dutch and English document.

One handicap in separating graphic design and writing text is that both the graphic designer and the author must make sure that they know where the information ends up. Graphic designers use professional drawing packages that authors don't have access to, or demand in–depth knowlegde of the application. Authors on the other hand know how to use TEX to typeset math, and drawing applications seldom provide proper support for math. Separating drawing the graphic and defining the labels also has the advantage that the labels can be typeset in a way that suits the document style (and specifically the fonts that are used).

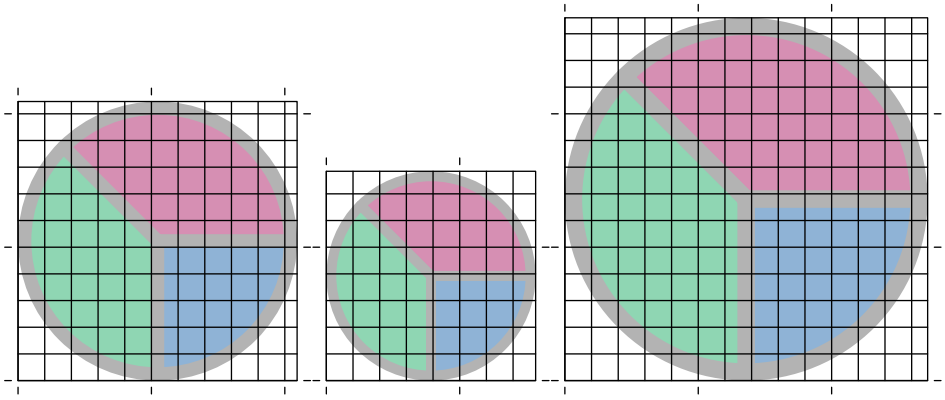
Although maintaining label specific data, like for instance the locations where labels have to end up, is possible as an independent activity, it may give the artist an uneasy feeling, especially because he is used to click and point tools. Therefore we will also discuss how to interface to Adobe Illustrator, a popular drawing application.

Grids

Imagine that we have the following graphic defined in a drawing program.

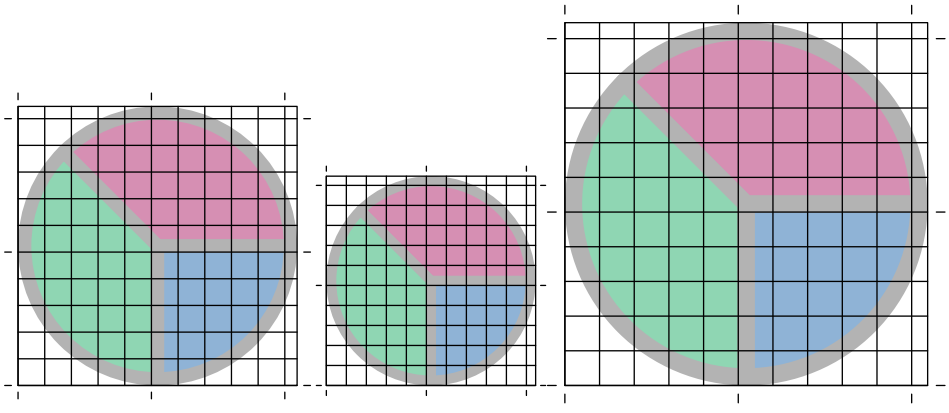


If you want to add some texts to this graphic, you need to know where these should be anchored. One way to achieve this is to put an imaginary grid on top of the graphic and anchor labels at fixed positions. Because graphics can be included at different sizes, such a grid may change accordingly. Imagine that you would have to define labels using the grids of the following graphics.



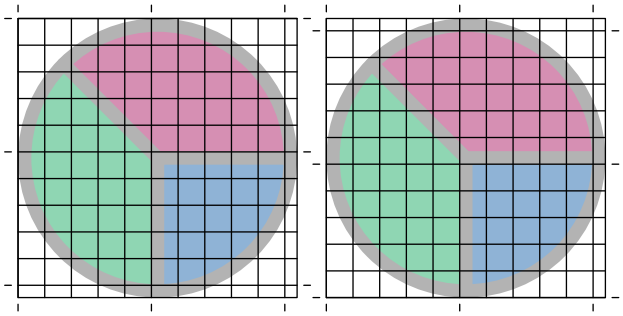
natural size scaled 75% scaled 130%

In practice you will define points on a fixed grid layed over the graphic scaled at 100%. In that case the grid will scale with the graphic.



natural size scaled 75% scaled 130%

Most drawing programs put their reference points in the lower left corner. This makes sense since that suits traditional coordinate systems. However, in a text flow it makes more sense to think top–down.



top anchored bottom anchored

The labeling mechanism described here works bottom up, which is opposite to the default top–down placement in ConTeXt text layers.

Adding text labels

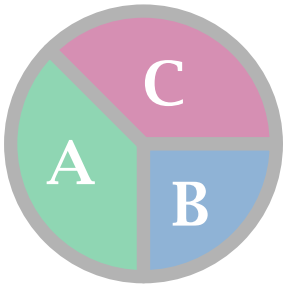
This is a preliminary description. Multiple language and label sets will be discussed as soon as we consider the interface stable. We will also support other interfaces and ways of positioning.

If you have to figure out the positions on your own, the following method can be used to add labels to a graphic.

```
\startfigurelabels[labels-1]
  \definefigurelabel[x=25bp,y=45bp]{\bfd\white A}
  \definefigurelabel[x=70bp,y=30bp]{\bfd\white B}
  \definefigurelabel[x=60bp,y=75bp]{\bfd\white C}
\stopfigurelabels
```

The graphic itself is placed in the usual way:

```
\startlinecorrection[blank]
\externalfigure[labels-1.mps][option=label]
\stoplinecorrection
```



or:

```
\placefigure
  {A floating figure}
  {\externalfigure[labels-1.mps][option=label,width=3cm]}
```

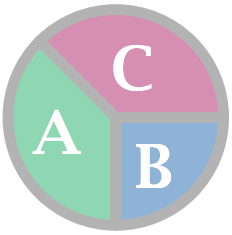


Figure 1 A floating figure

Although we limit ourselves here to simple labels, you can in principle put anything reasonable in a label.

Using symbolic positions

Especially when a graphic is used more than once with different labels, or when the task of defining the anchors can be delegated to the graphic designer, the separation between defining anchors and texts comes into view.

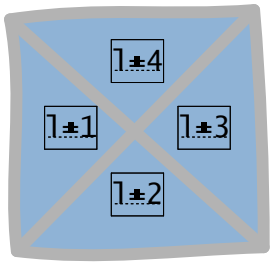
Each anchor gets a label (in its simplest form a number) and the positions are stored in a database. A record (which itself can be part of a figure (resource) library).

[testen: pos in fig database]

```
<rl:textlabels label="labels-2">
  <rl:textlabel label="1" x="25" y="50"/>
  <rl:textlabel label="2" x="50" y="25"/>
  <rl:textlabel label="3" x="75" y="50"/>
  <rl:textlabel label="4" x="50" y="75"/>
</rl:textlabels>
```

Here we have defined 4 positions that belong to figure `labels-2`.

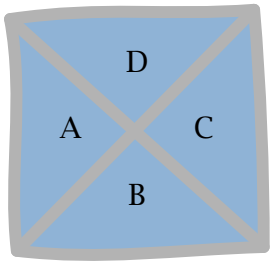
```
\startlinecorrection[blank]
\externalfigure[labels-2.mps][option=label]
\stoplinecorrection
```



It is possible to combine external and internal definitions, so you can use an external XML position database and define the label texts in the document itself. The label text definitions can be given in a TEX syntax or in XML.

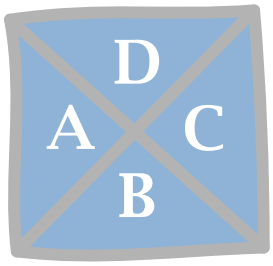
The database can also contain the text labels themselves, like:

```
<rl:textlabels label="labels-2">
  <rl:textlabel label="1">A</rl:textlabel>
  <rl:textlabel label="2">B</rl:textlabel>
  <rl:textlabel label="3">C</rl:textlabel>
  <rl:textlabel label="4">D</rl:textlabel>
</rl:textlabels>
```



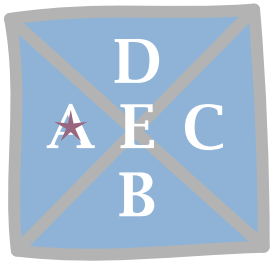
If the source document is a normal TEX document, you can include the definitions in your file.

```
\startfigurelabels[labels-2]
  \definefigurelabel[1]{\bfd\white A}
  \definefigurelabel[2]{\bfd\white B}
  \definefigurelabel[3]{\bfd\white C}
  \definefigurelabel[4]{\bfd\white D}
\stopfigurelabels
```



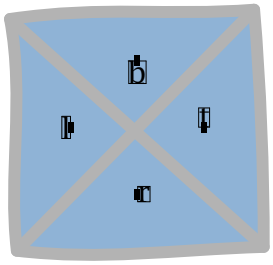
You can add additional labels. If needed you can provide your own coördinates.

```
\startfigurelabels[labels-2]
  \definefigurelabel[x=50bp,y=50bp]{\bfd\white E}
  \definefigurelabel[1]{\bfd\red\symbol[star]}
\stopfigurelabels
```

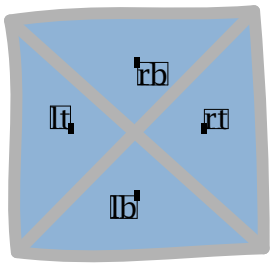


If you want a fresh start, you should explicitly reset the data with the reset command. We use this options to show you the alternative alignment locations (these are the same as in the ConTeXt layer mechanism).

```
\resetfigurelabels[labels-2]
\startfigurelabels[labels-2]
  \definefigurelabel[1][location=l]{l}
  \definefigurelabel[2][location=r]{r}
  \definefigurelabel[3][location=t]{t}
  \definefigurelabel[4][location=b]{b}
\stopfigurelabels
```



```
\resetfigurelabels[labels-2]
\startfigurelabels[labels-2]
  \definefigurelabel[1][location=lt]{lt}
  \definefigurelabel[2][location=lb]{lb}
  \definefigurelabel[3][location=rt]{rt}
  \definefigurelabel[4][location=rb]{rb}
\stopfigurelabels
```



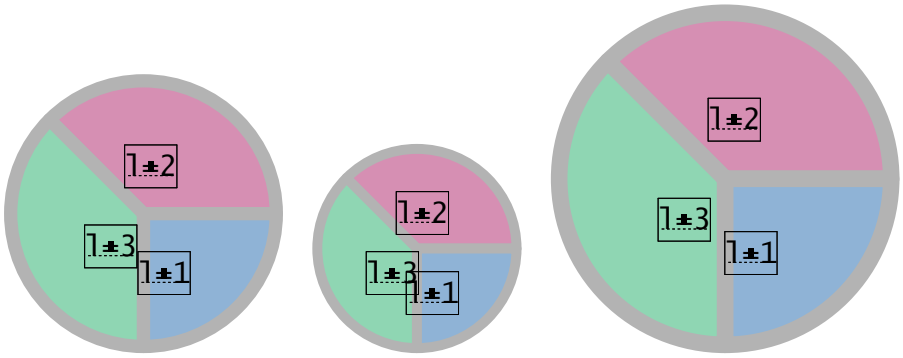
Adobe Illustrator files

The WARM plugin of Adobe Illustrator gives you the means to tag positions in a graphic. This plug–in is an initiative by Ross Moore and Wendy Mackay. The WARM plug–in writes special comment–only POSTSCRIPT files. This means that we don't have to ask graphic artists to use text base tools for providing the positional information.

The positions are normally numbered, but you can give them meaningful names. These positions are saved in files with the suffix `bb`. In Illustrator, these positions are named marked points. The data segment of such a file looks as follows:

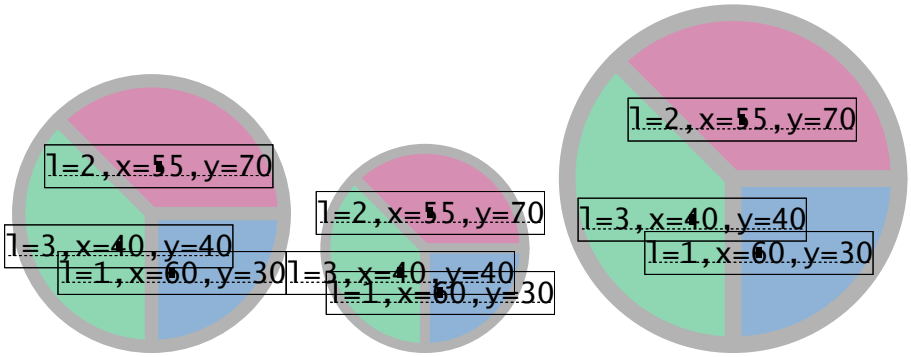
```
%%StartMarkedPoints
%%MarkedPoint: (60,30) : point(0,0) : 1 % Default Text
%%MarkedPoint: (55,70) : point(0,0) : 2 % Default Text
%%MarkedPoint: (40,40) : point(0,0) : 3 % Default Text
%%EndMarkedPoints
```

For our purpose, Only the coordinate (first entry) and label (third entry) make sense. There can best be some logic in placing the points, especially since we have to align the labels manually. When applied to our graphic, the previous definitions result in the following label anchors.



natural size scaled 75% scaled 125%

As you can see, the positions are scaled with the graphic, but the same bb is used for each of them.

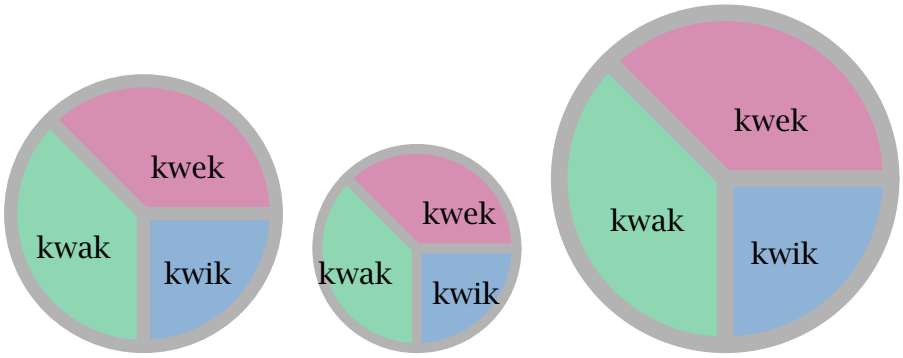


natural size scaled 75% scaled 125%

We define some labels:

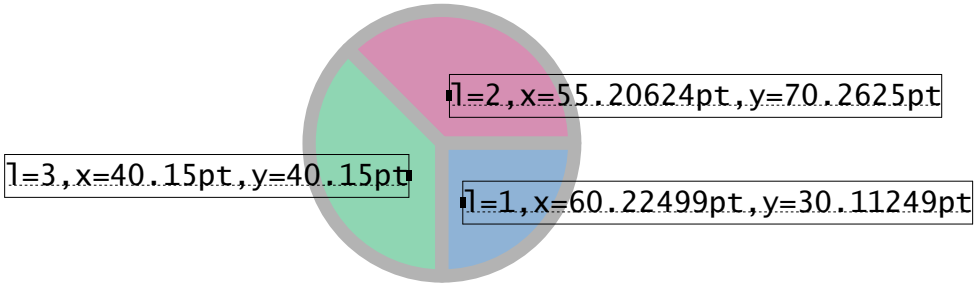
```
\startfigurelabels[labels-1]
  \definefigurelabel[1][location=r]{kwik}
  \definefigurelabel[2][location=r]{kwek}
  \definefigurelabel[3][location=l]{kwak}
\stopfigurelabels
```

These show up as follows. Watch how we aligned them left and right of the anchor point.



natural size scaled 75% scaled 125%

Internally, TEX works with real points, like `12pt`, but if needed you can define positions in POSTSCRIPT points, like `12bp`. Watch out: these are not the same, although for applications like these the difference does not show of that fast.

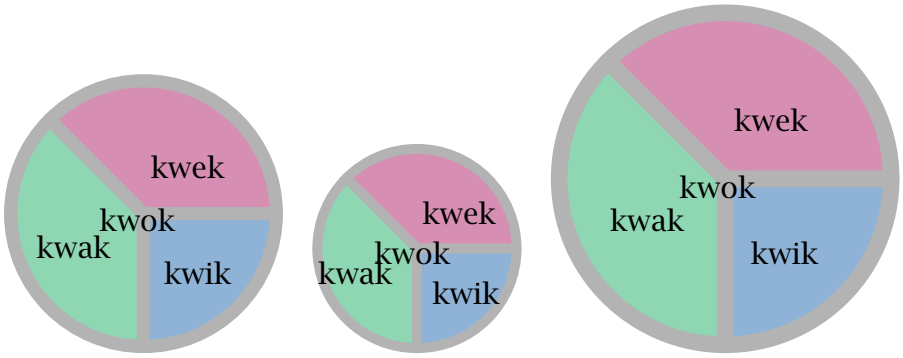


It is quite possible that the author wants to put a couple of extra labels in a graphic.

```
\startfigurelabels[labels-1]
  \definefigurelabel[x=50bp,y=50bp]{kwok}
\stopfigurelabels
```

The entries are added to the already defined ones; if you want to start fresh, you should explicitly reset the label texts with:

```
\resetfigurelabels[labels-1]
```



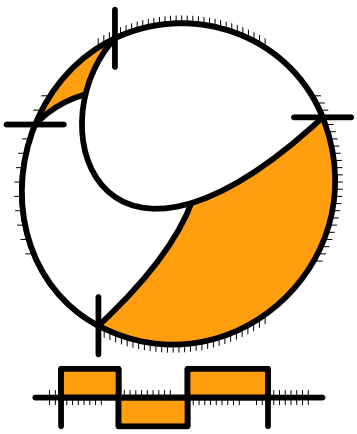
natural size scaled 75% scaled 125%

The WARM method is hooked into the external figure mechanism as (optional) second step in resolving layers. This means that an existing XML definition (file) takes precedence. In any case, the way to invoke this feature is the same:

```
\externalfigure[name][option=label]
```

In combination with previously defined labels this will give you labeled figures, given that a `bb` file is present. You can convert such files to an XML file using the `bbtoxml` PERL script. The generated base can be registered by saying:

```
\usefigurelabelbase[reset] \usefigurelabelbase[bbtoxml]
```



PRAGMA

Advanced Document Engineering | Ridderstraat 27 | 8061GH Hasselt NL
tel: +31 (0)38 477 53 69 | email: pragma@wxs.nl | internet: www.pragma-ade.com