The aamisc and aaobj MetaPost packages

Anders Andersen
aa@computer.org

Preface

This is an introduction to the two MetaPost packages aamisc and aaobj. They are used to draw figures with objects, interfaces, bindings and so on. They also provide more generic shapes like rectangles, lines, arrows, fillings. This note contains some examples of the most common macros available in these two packages. The appendix contains a short description of every macro available. The beginning of MetaPost files using these two packages usually look like this:

```
verbatimtex
%&latex
  \documentclass{article}
  \begin{document}
etex

input aamisc;
input aaobj;
```

Since the above verbatimtex block is LaTeX MetaPost has to be told that it should use LaTeX instead of TeX when processing TeX/LaTeX commands. This is achieved by the line "%&latex" (or by setting the TEX environment variable to latex). The examples following below use these predefined constants:

```
numeric width, height, orad;
width:=2cm; height:=1cm; orad:=0.5cm;
pair a, b, c, d;
a:=(0,2cm); b:=(0,0); c:=(3cm,2cm); d:=(3cm,0);
```

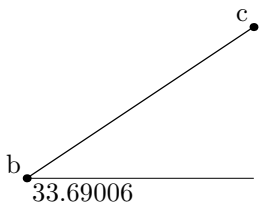
1 aamisc

The aamisc package is a collection of miscellaneous macros to ease the process of drawing (my) figures with MetaPost. It provides general macros for fillings, rectangles, lines and arrows. It also provides some more specialised macros for methods, attributes, buffers, threads, ques, resources and managers. The aaobj package use macros from the aamisc package.

1.1 Help functions

The anglebetween macro is used to calculate the direction (angle) between two points. Below is the direction from b to c calculated and printed.

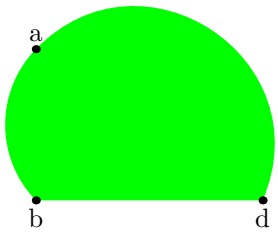
```
beginfig(1);

  draw b--c;
  draw b--d withpen pencircle scaled 0.2pt;

  dotlabel.ulft("b", b);
  dotlabel.ulft("c", c);
  label.lrt(decimal anglebetween(b,c), b);

endfig;
```



1.2 Fillings

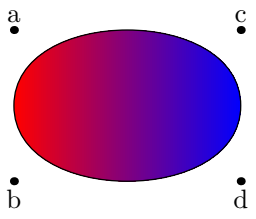
Any (cycled) path can be filled with any colour:

```
beginfig(2);

  path p;
  p:=a..b--d..cycle;

  draw fillpath(p, green);

  dotlabel.top("a", a);
  dotlabel.bot("b", b); dotlabel.bot("d", d);

endfig;
```



Gradient fillings (`fillcolor`) can be used to create some nice colour effects. The path to be filled has to be closed (a cycle) and the filling-corners (`a`, `b`, `c` and `d` below) have to be picked carefully. The `fillxcolor` are used to debug gradient fillings created with `fillcolor`. Problems and errors can be detected by the lines `fillxcolor` draws to illustrate how `fillcolor` works. Select a low gradient number (≈ 10) when you are using `fillxcolor` (the gradient number 100 below is to high).

```
beginfig(3);

  path p;
  p:=(a+b)/2..(a+c)/2..(c+d)/2..(d+b)/2..cycle;

  fillcolor(p, 100, a, c, b, d, red, blue);
  draw p;

  dotlabel.top("a", a); dotlabel.top("c", c);
  dotlabel.bot("b", b); dotlabel.bot("d", d);

endfig;
```

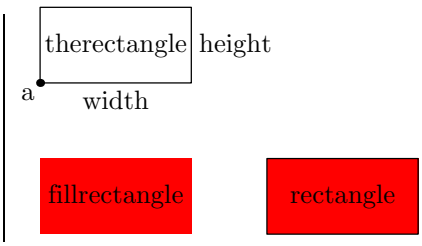


1.3 Rectangles

Most of the shapes provided by `aamisc` (and `aaobj`) have three different versions: a path (`the...`), a filling (`fill...`) and the complete shape with the path and the (optional) filling (`...`). Macros of the first type returns a MetaPost path. The second type returns a MetaPost picture. The last type with the optional filling is either a MetaPost picture (with filling) or a MetaPost path

(without filling). The three `draw` statements below illustrate these three different versions of the rectangle.

```
beginfig(4);

  draw therectangle(a, width, height);
  label("therectangle", a+(0.5width,0.5height));
  dotlabel.llft("a", a);
  label.bot("width", a+(0.5width,0));
  label.rt("height", a+(width,0.5height));

  draw fillrectangle(b, width, height, red);
  label("fillrectangle", b+(0.5width,0.5height));
  draw rectangle.red(d, width, height);
  label("rectangle", d+(0.5width,0.5height));

endfig;
```



The difference between the three rectangle macros above and the center rectangle macros are the meaning of the first argument, the position. The first argument of the center rectangle is the position of the center of the rectangle, while the positions in the examples above are the lower left corner of the rectangle (if the next two arguments are positive). A rectangle with rounded corners is also available. It has an extra argument that gives the radius of the $\frac{1}{4}$ of a circle in each corner. See the appendix for details.

1.4 Methods and attributes

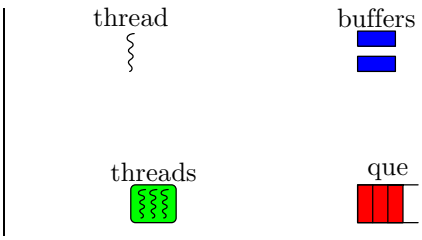
This is a way to draw methods (functions) and attributes, usually inside an object. The methods are illustrated as sheets with lines (code) on and the attributes are just dots.

```
beginfig(5);

  draw method.red("m", a-(4mm,0), 8mm, 4, 2mm);
  draw attr.blue("a", b, 2mm);

endfig;
```

1.5 Buffers, threads and ques

Buffers, threads (thread pools) and ques are drawn as shown below.

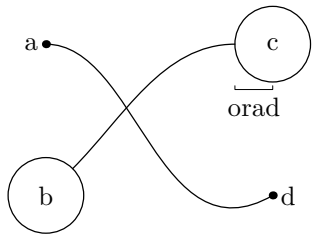
```
beginfig(6);

  draw thethread(a, 5mm, 1.5mm);
  label.top("thread", a+(0,5mm));
  draw threads.green("threads", b, 3, 1.5mm, 5mm);

  draw buffers.blue("buffers", c, 5mm, 2, 2mm);
  draw que.red("que", d, 3, 2mm, 5mm);

endfig;
```



1.6 Lines and arrows

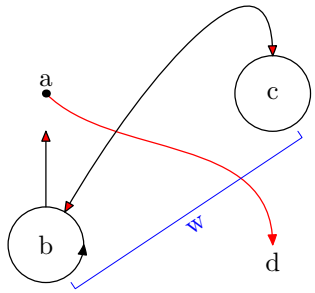
A set of macros to draw lines between to points with a given start and a given stop directions are provided in `aamisc`. The example below also contains a line with cut-off at a given radius from the start and stop points (the line from `b` to `c`).

```
beginfig(7);

  draw dirline(a, 0, 30, d);
  dotlabel.lft("a", a); dotlabel.rt("d", d);

  draw fullcircle scaled 2orad shifted b;
  draw fullcircle scaled 2orad shifted c;
  label("b", b); label("c", c);
  draw ddirline(b, 45, 0, c, orad);
  draw lengthmarker("orad",
    c-(orad,orad), c-(0,orad));

endfig;
```



The arrow macros are very similar to the line macros above. The `aamisc` package also provides a length-marker (see both above and below).

```
beginfig(8);

  draw dirarrow.red(a, -45, -90, d) withcolor red;
  dotlabel.top("a", a); label.bot("d", d);

  draw arrow(fullcircle scaled 2orad shifted b);
  draw fullcircle scaled 2orad shifted c;
  label("b", b); label("c", c);
  draw ddirdblarrow.red(b, 60, -90, c, orad);
  draw darrow.red(b, a, orad);

  pair mb, mc;
  mb:=(b-(0,orad)) rotatedaround(b,
    anglebetween(b, c));
  mc:=(c-(0,orad)) rotatedaround(c,
    anglebetween(b, c));
  draw lengthmarker("w", mb, mc) withcolor blue;

endfig;
```



1.7 Loops

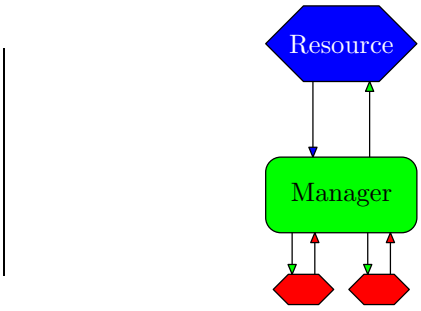
A loop and a timed (clock synchronised) loop are used to illustrate looping (repeating) activities (processes).

```
beginfig(9);

  draw loop(a, 3mm);
  draw timedloop(c, 3mm);

endfig;
```

1.8 Resources and managers

Abstract resources and their managers and providers are easy to draw with the `aamisc` package. A set of macros to draw their relations (arrows) are also provided.

```
beginfig(10);

  pair x, y;
  x:=(-.5cm,-1.25cm); y:=(.5cm,-1.25cm);

  draw resource.blue(a, 1cm);
  draw thelabel("Resource", a) withcolor white;
  draw provides.green(b, a, 1cm, 1cm);
  draw providedby.blue(a, b, 1cm, 1cm);

  draw manager.green(b, 2cm, 1cm);
  label("Manager", b);

  draw resource.red(x, .4cm);
  draw manages.green(b, x, 1cm, .4cm);
  draw managedby.red(x, b, .4cm, 1cm);
  draw resource.red(y, .4cm);
  draw manages.green(b, y, 1cm, .4cm);
  draw managedby.red(y, b, .4cm, 1cm);

endfig;
```

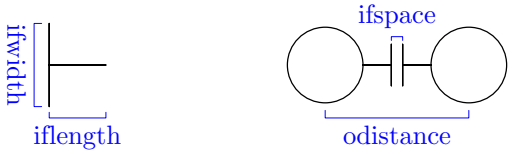


2 aaobj

The `aaobj` package contains macros to draw objects (components), their interfaces and different kinds of bindings.

2.1 Help values

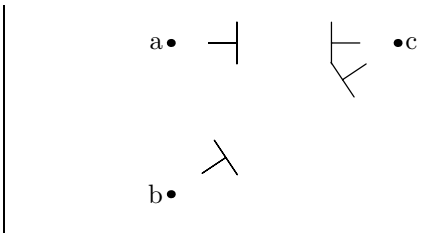
The following constants are defined (as a factor of the object radius): `iflength` is the length of an interface, `ifwidth` is the width of an interface, `ifspace` is the space between two (connected) interfaces, and `odistance` is the distance between two objects directly connected with interfaces.



2.2 Interfaces

We have two ways of drawing interfaces, either as an interface connected to (an object at) a given position and pointing in one of eight possible directions (`theif`) or as an interface connected to (an object at) a given position and pointing at another position or another object (`theiffromto`). These two macros returns a path. The picture versions of these macros are called `anif` and `aniffromto`, respectively. The example below illustrate these four macros.

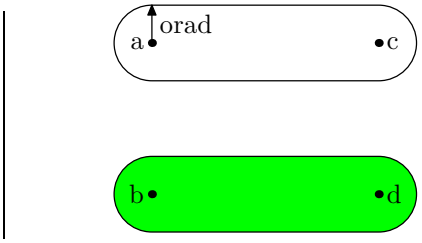
```
beginfig(12);

  draw theif.rt(a, orad);
  dotlabel.lft("a", a);

  draw theiffromto(b, c, .5cm);
  dotlabel.lft("b", b);

  dotlabel.rt("c", c);
  draw anif.lft(c, orad);
  draw aniffromto(c, b, orad);

endfig;
```



2.3 Bindings from-to

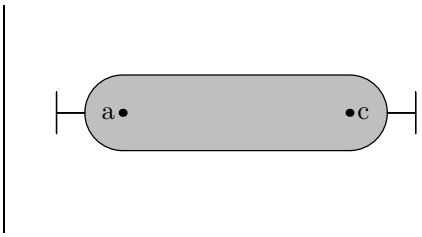
The to first arguments in the binding from-to macros are the endpoints within the binding. The last argument is the radius (width) of the binding. The example below illustrate this.

```
beginfig(13);

  draw thebindingfromto(a, c, orad);
  dotlabel.lft("a", a); dotlabel.rt("c", c);
  drawarrow a--a+(0,orad);
  label.rt("orad", a+(0,0.5orad));

  draw bindingfromto.green(b, d, orad);
  dotlabel.lft("b", b); dotlabel.rt("d", d);

endfig;
```



Another example where the interfaces to the binding is included and a non-standard colour is selected.

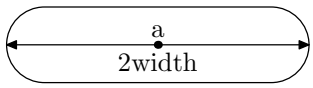
```
beginfig(14);

  color grey;
  grey:=(0.75, 0.75, 0.75);

  draw bindingfromto.grey(a, c, orad);
  draw theif.lft(a, orad);
  draw theif.rt(c, orad);
  dotlabel.lft("a", a); dotlabel.rt("c", c);

endfig;
```



2.4 Bindings

These macros have three arguments (and a possible suffix): position or center of binding, radius (as above), and length. The example below illustrate the use of one of these binding macros.

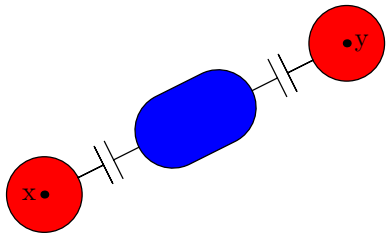
```
beginfig(15);

  draw binding(a, orad, 2width);
  dotlabel.top("a", a);
  drawarrow a-(0,0)--a-(width,0);
  drawarrow a-(0,0)--a+(width,0);
  label.bot("2width", a);

endfig;
```



2.5 Bindings between

These macros are used to draw bindings between objects when the position of the objects are known. The example below illustrates a binding between objects at positions `x` and `y`.

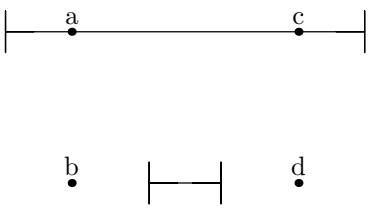
```
beginfig(16);

  pair x, y;
  x:=(0,0); y:=(4cm,2cm);

  draw bindingbetween.blue(x, y, orad);

  draw object.red(x, orad);
  draw theiffromto(x, y, orad);
  draw object.red(y, orad);
  draw theiffromto(y, x, orad);

  dotlabel.lft("x", x); dotlabel.rt("y", y);

endfig;
```



2.6 Implicit bindings

An implicit binding is just two interfaces connected with a line. Implicit bindings are also available in three different versions: binding from-to, bindings, and binding between.

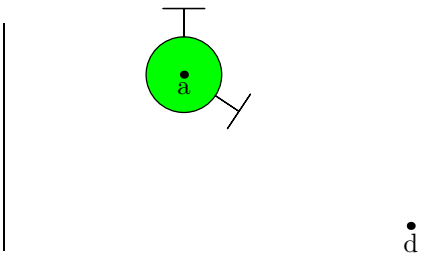
```
beginfig(17);

  draw theimpbindingfromto(a, c, orad);
  dotlabel.top("a", a); dotlabel.top("c", c);

  draw theimpbindingbetween(b, d, orad);
  dotlabel.top("b", b); dotlabel.top("d", d);

endfig;
```



2.7 Objects

An object is drawn as a circle. It usually has some interfaces associated with it.

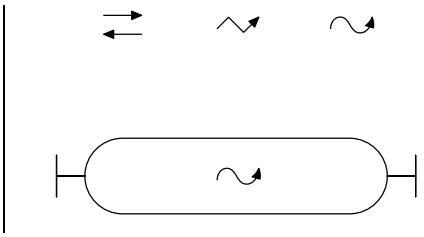
```
beginfig(18);

  draw object.green(a, orad);
  draw theif.top(a, orad);
  draw theiffromto(a, d, orad);

  dotlabel.bot("a", a); dotlabel.bot("d", d);

endfig;
```



2.8 Binding types

We can mark a binding with three different type-marks: operational, signal and stream.

```
beginfig(19);

  draw opbindingmark(a, orad);
  draw sigbindingmark((a+c)/2, orad);
  draw streambindingmark(c, orad);

  draw binding((b+d)/2, orad, 2width);
  draw theif.lft(b, orad);
  draw theif.rt(d, orad);
  draw streambindingmark((b+d)/2, orad);

endfig;
```



A Complete macro listing

Below follows a complete listing of all macros provided by the `aamisc` and `aaobj` MetaPost macro packages. Underlined macros (like `rectangle`) are the most common one and most users can survive with these for a long a time. The suffix of all the macros below is an optional argument. For example the `col` suffix of the `rectangle` macro below is optional. The result without this suffix is the path of a rectangle. The result with this suffix is a picture containing a rectangle with a filling in the given colour.

A.1 `aamisc`

A.1.1 Help functions

`anglebetween(apos, bpos)`: Calculate the direction in degrees from `apos` to `bpos`.

A.1.2 Fillings

`fillpath(p, col)`: Fill the path `p` with the colour `col`

`pathwithfill.col(p)`: Draw the path `p` and optionally fill it with colour `col`.

`fillcolor(p, nc, astart, astop, bstart, bstop, cstart, cstop)`: Gradient colour filling.

`fillxcolor(p, nc, astart, astop, bstart, bstop, cstart, cstop)`: A help function.

A.1.3 Rectangles

`therectangle(pos, x, y)`: A rectangle with lower left corner at `pos`, width `x` and height `y`.

`fillrectangle(pos, x, y, col)`: Filling of rectangle above with colour `col`.

`rectangle.col(pos, x, y)`: Rectangle above with optional colour filling `col`.

`thecrectangle(pos, x, y)`: A rectangle with center at `pos`, width `x` and height `y`.

`fillcrectangle(pos, x, y, col)`: Filling of rectangle above with colour `col`.

`crectangle.col(pos, x, y)`: Rectangle above with optional colour filling `col`.

`theroundrec(pos, x, y, d)`: A rounded rectangle with corner radius `d`.

`fillroundrec(pos, x, y, d, col)`: Filling of rounded rectangle above with colour `col`.

`roundrec.col(pos, x, y)`: Rounded rectangle above with optional colour filling `col`.

`thecroundrec(pos, x, y, d)`: A centered rounded rectangle with corner radius `d`.

`fillcroundrec(pos, x, y, d, col)`: Filling of rounded rectangle above with colour `col`.

`croundrec.col(pos, x, y, d)`: Rounded rectangle above with optional colour filling `col`.

A.1.4 Methods and attributes

`method.col(name, pos, x, ny, dy)`: A method `name` with lower left corner at `pos` and `ny` lines.

`cmethod.col(name, pos, x, ny, dy)`: A method `name` with center at `pos` and `ny` lines.

`attr.col(name, pos, rad)`: An attribute `name` at `pos` with radius `rad`.

A.1.5 Buffers, threads and ques

`buffers.col(name, pos, x, ny, dy)`: Buffer `name` at `pos` with `ny` elements.

`thethread(pos, len)`: A thread of length `len` at `pos`.

`threads.col(name, pos, nx, dx, y)`: A thread-pool `name` at `pos` with `nx` threads.

`que.col(name, pos, nx, dx, y)`: A que `name` at `pos` with `nx` elements.

A.1.6 Lines and arrows

`dirline(apos, adir, bdir, bpos)`: A line from `apos` to `bpos` in given directions.

`ddirline(apos, adir, bdir, bpos, d)`: A line as above with cut-off distance `d`.

`arrow.col(p)`: An arrow following the path `p`.

`darrow.col(apos, bpos, d)`: An arrow from `apos` to `bpos` with cut-off distance `d`.

`dirarrow.col(apos, adir, bdir, bpos)`: An arrow from `apos` to `bpos` in given directions.

`ddirarrow.col(apos, adir, bdir, bpos, d)`: An arrow as above with cut-off distance `d`.

`dblarrow.col(p)`: A two-way arrow following the path `p`.

`ddblarrow.col(apos, bpos, d)`: A two-way arrow from `apos` to `bpos` with cut-off distance `d`.

`dirdblarrow.col(apos, adir, bdir, bpos)`: An two-way arrow from `apos` to `bpos`.

`ddirdblarrow.col(apos, adir, bdir, bpos, d)`: An two-way arrow as above with cut-off.

`lengthmarker.inv(name, apos, bpos)`: A length marker between `apos` and `bpos`.

A.1.7 Loops

`loop(pos, rad)`: A loop at `pos` with radius `rad`.

`timedloop(pos, rad)`: A timed loop at `pos` with radius `rad`.

A.1.8 Resources and managers

`thresource(pos, size)`: A resource at position `pos` and size (height) `size`.

`fillresource(pos, size, col)`: Filling of resource above with colour `col`.

`resource.col(pos, size)`: Resource above with optional colour filling `col`.

`themanager(pos, width, height)`: A manager at position `pos` and size `width`×`height`.

`fillmanager(pos, width, height, col)`: Filling of manager above with colour `col`.

`manager.col(pos, width, height)`: Manager above with optional colour filling `col`.

`provides(from, to, fsiz, tsiz)`: A manager at `from` provides resource at `to`.

`providedby(from, to, fsiz, tsiz)`: A resource at `from` is provided by manager at `to`.

`manages(from, to, fsiz, tsiz)`: A manager at `from` manages resource at `to`.

`managedby(from, to, fsiz, tsiz)`: A resource at `from` is managed by manager at `to`.

A.2 aaobj

A.2.1 Help values

`iflength`: The length of an interface as a factor of `rad` (the object radius).

`ifwidth`: The width of an interface as a factor of `rad`.

`ifspace`: The space between two interfaces as a factor of `rad`.

`odistance`: The space between two objects directly connected with interfaces.

`halfbinding.inv(apos, bpos, rad)`: A help macro used by the binding macros.

A.2.2 Interfaces

`theif.dir(pos, rad)`: An interface in direction `dir` for object at position `pos` with radius `rad`.

`anif.dir(pos, rad)`: An interface as above, but this is a picture (not a path).

`theiffromto(apos, bpos, rad)`: An interface at position `apos` pointing at position `bpos`.

`aniffromto(apos, bpos, rad)`: An interface as above, but this is a picture (not a path).

A.2.3 Bindings from-to

`thebindingfromto(apos, bpos, rad)`: A binding from `apos` to `bpos` with radius `rad`.

`fillbindingfromto(apos, bpos, rad, col)`: Filling of binding above with colour `col`.

`bindingfromto.col(apos, bpos, rad)`: Binding above with optional colour filling `col`.

A.2.4 Bindings

`thebinding(pos, rad, length)`: A binding at `pos` with radius `rad` and length `length`.

`fillbinding(pos, rad, length, col)`: Filling of binding above with colour `col`.

`binding.col(pos, rad, length)`: Binding above with optional colour filling `col`.

A.2.5 Bindings between

`thebindingbetween(apos, bpos, rad)`: A binding between `apos` and `bpos` with radius `rad`.

`fillbindingbetween(apos, bpos, rad, col)`: Filling of binding above with colour `col`.

`bindingbetween.col(apos, bpos, rad)`: Binding above with optional colour filling `col`.

A.2.6 Implicit bindings

`theimpbindingfromto(apos, bpos, rad)`: An implicit binding from `apos` to `bpos`.

`theimpbinding(pos, rad, length)`: An implicit binding at `pos` with length `length`.

`theimpbindingbetween(apos, bpos, rad)`: A implicit binding between `apos` and `bpos`.

A.2.7 Objects

`theobject(pos, rad)`: An object at position `pos` and with radius `rad`.

`fillobject(pos, rad, col)`: Filling of object above with colour `col`.

`object.col(pos, rad)`: Object above with optional colour filling `col`.

A.2.8 Binding types

`opbindingmark.deg(pos, rad)`: An operational binding mark.

`invopbindingmark.deg(pos, rad)`: An operational binding mark in opposite direction.

`sigbindingmark.deg(pos, rad)`: A signal binding mark.

`invsigbindingmark.deg(pos, rad)`: A signal binding mark in opposite direction.

`streambindingmark.deg(pos, rad)`: A stream binding mark.

`invstreambindingmark.deg(pos, rad)`: A stream binding mark in opposite direction.

B Resources

The `aamisc` and `aaobj` MetaPost packages are available from the following location:

`http://www.cs.uit.no/~aa/dist/texmf/mp/aaobj/`

MetaPost usually ships with your TEX distribution, but more information is available at the following location:

`http://cm.bell-labs.com/who/hobby/MetaPost.html`