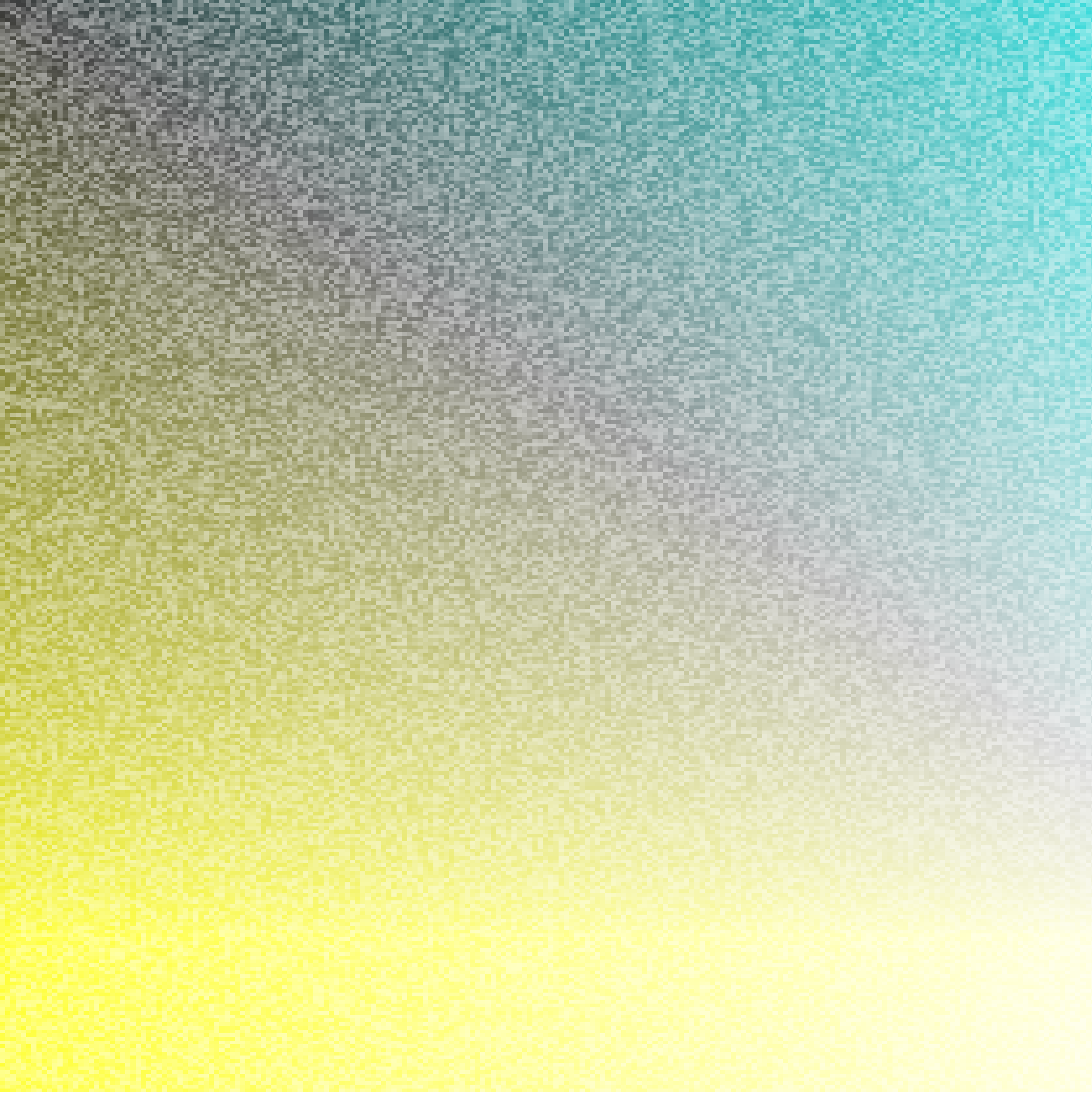

Graphics

Hans Hagen

Introduction

This manual is about integrating graphics your document. Doing this is not really that complex so this manual will be short. Because graphic inclusion is related to the backend some options will discussed. It’s typical one of these manuals that can grow over time.

Inclusion

The command to include an image is:

```
                    1     2         3
\externalfigure [...] [...] [..,..=..,..]
                       OPT       OPT
1  FILE

2  NAME

3  inherits: \setupexternalfigure
```

and its related settings are:

```
                          1           2
\setupexternalfigure [...,...] [..,..=..,..]
                          OPT
1  NAME

2  width               =  DIMENSION
   height              =  DIMENSION
   label               =  NAME
   page                =  NUMBER
   object              =  yes no
   prefix              =  TEXT
   method              =  pdf mps jpg png jp2 jbig svg eps gif tif mov buffer tex cld auto
   controls            =  yes no
   preview             =  yes no
   mask                =  none
   resolution          =  NUMBER
   color               =  COLOR
   arguments           =  TEXT
   repeat              =  yes no
   factor              =  fit broad max auto default
   hfactor             =  fit broad max auto default
   wfactor             =  fit broad max auto default
   maxwidth            =  DIMENSION
   maxheight           =  DIMENSION
   equalwidth          =  DIMENSION
   equalheight         =  DIMENSION
   scale               =  NUMBER
   xscale              =  NUMBER
   yscale              =  NUMBER
   s                   =  NUMBER
   sx                  =  NUMBER
   sy                  =  NUMBER
   lines               =  NUMBER
   location            =  local global default
   directory           =  PATH
   option              =  test frame empty
   forgroundcolor      =  COLOR
```

```
    reset                    =  yes no
    background               =  color foreground NAME
    frame                    =  on off
    backgroundcolor          =  COLOR
    xmax                     =  NUMBER
    ymax                     =  NUMBER
    frames                   =  on off
    interaction              =  yes all none reference layer bookmark
    bodyfont                 =  DIMENSION
    comment                  =  COMMAND TEXT
    size                     =  none media crop trim art
    cache                    =  PATH
    resources                =  PATH
    display                  =  FILE
    conversion               =  TEXT
    order                    =  LIST
    crossreference           =  yes no NUMBER
    transform                =  auto NUMBER
    userpassword             =  TEXT
    ownerpassword            =  TEXT
```

So you can say:

```
\externalfigure[cow.pdf][width=4cm]
```

The suffix is optional, which means that this will also work:

```
\externalfigure[cow][width=4cm]
```

Defining

todo

```
\useexternalfigure [.1.] [.2.] [.3.] [..,..=4..,..]
                                 OPT        OPT
1  NAME
2  FILE
3  NAME
4  inherits: \setupexternalfigure
```

```
\defineexternalfigure [.1.] [.2.] [..,..=3..,..]
                            OPT        OPT
1  NAME
2  NAME
3  inherits: \setupexternalfigure
```

```
\registerexternalfigure [.1.] [.2.] [..,..=3..,..]
                              OPT        OPT
1  FILE
2  NAME
3  inherits: \setupexternalfigure
```

Analyzing

todo

```
\getfiguredimensions [...] [..,..=..,..]
                      1          2
                                OPT
1  FILE
2  inherits: \setupexternalfigure
```

```
\figurefilename
```

```
\figurefilepath
```

```
\figurefiletype
```

```
\figurefullname
```

```
\figureheight
```

```
\figurenaturalheight
```

```
\figurenaturalwidth
```

```
\figuresymbol [...] [..,..=..,..]
               1          2
                         OPT
1  FILE NAME
2  inherits: \externalfigure
```

```
\figurewidth
```

```
\noffigurepages
```

Collections

todo

```
\externalfigurecollectionmaxheight {...*}
* NAME
```

```
\externalfigurecollectionmaxwidth {...*}
* NAME
```

```
\externalfigurecollectionminheight {...*}
* NAME
```

```
\externalfigurecollectionminwidth {...*}
* NAME
```

```
\externalfigurecollectionparameter {...1} {...2}
1 NAME
2 KEY
```

```
\startexternalfigurecollection [...*] ... \stopexternalfigurecollection
* NAME
```

Conversion

todo

Figure databases

todo

```
\usefigurebase [...*]
* reset FILE
```

Overlays

todo

```
\overlayfigure {...}
                 *
*   FILE
```

```
\pagefigure [...] [..,..=..,..]
              1        2
                      OPT
1   FILE
2   offset     =  default overlay none DIMENSION
```

Scaling

Images are normally scaled proportionally but if needed you can give an explicit height and width. The `\scale` command shares this property and can be used to scale in the same way as `\externalfigure`. I will illustrate this with an example.

You can define your own bitmaps, like I did with the cover of this manual:

```
\startluacode

    local min, max, random = math.min, math.max, math.random

    -- kind of self-explaining:

    local xsize      = 210
    local ysize      = 297
    local colordepth = 1
    local usemask    = true
    local colorspace = "rgb"

    -- initialization:

    local bitmap = graphics.bitmaps.new(xsize,ysize,colorspace,colordepth,usemask)

    -- filling the bitmap:

    local data    = bitmap.data
    local mask    = bitmap.mask
    local minmask = 100
    local maxmask = 200

    for i=1,ysize do
        local d = data[i]
        local m = mask[i]
        for j=1,xsize do
            d[j] = { i, max(i,j), j, min(i,j) }
            m[j] = random(minmask,maxmask)
        end
    end

    -- flushing the lot:
```

```
    graphics.bitmaps.tocontext(bitmap)

\stopluacode
```

The actually inclusion of this image happened with:

```
\scale
  [width=\paperwidth]
  {\getbuffer[image]}
```

The backend

Traditionally TEX sees an image as just a box with dimensions and in LuaTEX it is actually a special kind of rule that carries information about what to inject in the final (pdf) file. In regular LuaTEX the core formats `pdf`, `png`, `jpg` and `jp2` are dealt with by the backend but in ConTEXt we can use Lua instead. We might default to that method at some point but for now you need to enable that explicitly:

```
\enabledirectrive[graphics.pdf.uselua]
\enabledirectrive[graphics.jpg.uselua]
\enabledirectrive[graphics.jp2.uselua]
\enabledirectrive[graphics.png.uselua]
```

All four can be enabled with:

```
\enabledirectrive[graphics.uselua]
```

Performance-wise only png inclusion can be less efficient, but only when you use interlaced images or large images with masks. It makes no real sense in a professional workflow to use the (larger) interlaced images, and masks are seldom used at high resolutions, so in practice one will not really notice loss of performance.

The advantage of this method is that we can provide more options, intercept bad images that make the backend abort and lessen the dependency on libraries.