

Playing with Flash in ConT_EXt-mkiv

Luigi Scarso

Email luigi.scarso@gmail.com
Address Padova,
Italy.

Abstract This describes a first attempt to adapt `flashmovie.sty` to ConT_EXt-mkiv to produce a flash movie with METAPOST and swftools.

1 Introduction

Beginning with release 9, AdobeReader comes with an embedded Flash player. The recent addition to CTAN of the `flashmovie` package by Timo Hartmann prompted me to investigate the feasibility of an integration between ConT_EXt-mkiv and Flash by using `pdf2swf`¹. In these experiments, all tests were performed under Linux Ubuntu 8.04 with AdobeReader 9.2 installed. Testing was not done on Windows or Mac systems, but it is likely these techniques will work there as well. This article assumes that readers are familiar with ConT_EXt-mkiv (more information is available at http://wiki.contextgarden.net/ConTeXt_Minimals).

2 First step: Adapting `flashmovie.sty`

Quoting from CTAN (`flashmovie.sty`) . . . *allows direct embedding of flash movies into PDF files. It is designed for use with `pdflatex`. The package takes advantage of the embedded Adobe Flash player in Adobe Reader 9; the reader is invoked with the ‘rich media annotation’ feature, described in “Adobe Supplement to the ISO 32000 BaseVersion: 1.7 ExtensionLevel: 3”. This method of embedding movies is attractive since it removes all platform dependencies; however, the user is required*

¹. `pdf2swf` is a program from <http://www.swftools.org> that is based on xpdf codebase.

to use Acrobat 9. `flashmovie.sty` is a simple style, so there is almost nothing to change for use with ConTeXt-mkiv: just add a suffix `ctx` to the original `\flashmovieembedfile` and `\flashmovie` macros, and save them in `supp-swf.mkivs`.

```
%D \module
%D [      file=supp-swf,
%D      version=2009.12.31,
%D      title=\CONTEXT\ Support Macros,
%D      subtitle=swf inclusion,
%D      author=Luigi Scarso,
%D      date=\currentdate,
%D      copyright={Luigi Scarso}]

\unprotect

\newcount\filespecnum
\newcount\configurationnum
\newcount\contentnum
\newcount\settingsnum
\newdimen\xxwidth
\newdimen\xxheight
\newbox\xxcontent

\def\flashmovieembedfilectx#1{
\immediate
\pdfobj stream
  attr { /Type/EmbeddedFile }
  file {#1}
\immediate
\pdfobj { <<
  /Type /Filespec
  /F (#1)
  /UF (#1)
  /EF << /F \the\pdflastobj\space 0 R >>
  >>}}

\def\flashmoviectx[#1]{%
\getparameters[flashmovieparams][width=4cm,height=4cm,#1]
```

```

\flashmovieembedfilectx{\csname flashmovieparamsfile\endcsname}
\filespecnum=\pdflastobj
\immediate
\pdfobj
  {<<
    /Instances
    [<<
      /Asset \the\filespecnum\space 0 R
      /Params << /Binding /Foreground >>
    >>]
    /Subtype /Flash
  >>}
\configurationnum=\pdflastobj

\immediate
\pdfobj
  {<<
    /Assets << /Names [(\csname flashmovieparamsfile\endcsname)
      \the\filespecnum\space 0 R] >>
    /Configurations [\the\configurationnum\space 0 R]
  >>}
\contentnum=\pdflastobj
\immediate
\pdfobj
  {<<
    /Activation
    << /Type /RichMediaActivation
      /Condition /PO
      /Configuration \the\configurationnum\space 0 R
      /Animation
      << /Subtype /Linear
        /Speed 1
        /Playcount 1
      >>
    /Presentation
    << /PassContextClick false
      /Style /Embedded
      /Toolbar false
      /NavigationPane false
      /Transparent true
  >>}

```

```

    /Window
    << /Type /RichMediaWindow
    /Width << /Default 100 /Min 100 /Max 100 >>
    /Height << /Default 100 /Min 100 /Max 100 >>
    /Position
    << /Type /RichMediaPosition
    /HAlign /Near
    /VAlign /Near
    /HOffset 0
    /VOffset 0
    >>
  >>
>>
/Deactivation
<< /Type /RichMediaDeactivation
/Condition /XD
>>
>>}
\settingsnum=\pdflastobj
\setbox\xxcontent=\hbox to \csname flashmovieparamwidth\endcsname{%
\hss\vbox to\csname flashmovieparamsheight\endcsname{%
\hsize=\csname flashmovieparamwidth\endcsname\vss \vss }\hss}%
\xxwidth=\wd\xxcontent%
\xxheight=\ht\xxcontent%
\immediate\write16{wd=\the\xxwidth,ht=\the\xxheight,\the\textwidth}%
\box\xxcontent%
\pdfannot width \csname flashmovieparamwidth\endcsname height %
\csname flashmovieparamsheight\endcsname depth Opt {%
/Subtype /RichMedia
/RichMediaContent \the\contentnum\space 0 R
/RichMediaSettings \the\settingsnum\space 0 R
}}
\protect

```

To test this, visit the site <http://melusine.eu.org/syracuse/metapost/animations/chupin/?idsec=filtre> and download the sample file filtre-num.swf. Use this file as follows:

```

%%test-ctx.tex
\input supp-swf.mkiv
\starttext
\flashmoviectx[width=\textwidth,height=\textwidth,file=filtre-num.swf]
\stoptext

```

Remember that in ConT_EXt-mkiv a pdf is made with:

```
#>context test-ctx
```

3 Integrating Metapost and swftools

Nearly everything needed for METAPOST and Flash can be found at the nicely designed site <http://melusine.eu.org/syracuse>. Given that I now have a way to manage swf files in ConT_EXt-mkiv, the next step is to build a primitive system which imitates some of these beautiful animations with METAPOST.

I decided to use the ``ConT_EXt-mkii'' (aka ``external'') method:

1. save a "metapost animation sequence" in an external file;
2. run an external mpost on it;
3. convert the results file in one pdf by using another external ConT_EXt-mkiv run;
4. convert the pdf in swf with an external call to pdf2swf.

This method *does not* use the full power of LuaT_EX and ConT_EXt-mkiv: for example ConT_EXt-mkiv comes with a METAPOST interpreter embedded, but it cannot be used because an external mpost program is being called. Also, I do not extend luatex with a possible Lua binding to a libpdf2swf.so, simply because libpdf2swf.so is to be built entirely. So this method of porting it to ConT_EXt-mkii is simpler, but the fun here is to use the Lua language.

I split a "metapost animation sequence" between the preamble and body; the preamble can be shared with other bodies, and the body is enclosed in a for loop starting from start to end_limit with step step_value:

```

for frame:=start step step_value until end_limit :
  beginfig(frame)
  <body here>
  endfig;
end.

```

Let's continue with the `supp-swf.mkiv` module. The first step is to manage the creation of METAPOST files for animation. Each frame is a METAPOST figure n .mps which is saved in a name (default:out-mps) folder.

```

%% A lua table to collect animations
\ctxlua{MetapostFramesTable = {}}

%% animation preamble
\long\def\startPreambleMetapostFrames[#1]#2\stopPreambleMetapostFrames{%
\getparameters[preamblemps.] [name={out-mps},#1,body={#2}]
\ctxlua{%
MetapostFramesTable[tostring("\csname preamblemps.name\endcsname")] =
      tostring("\csname preamblemps.body\endcsname")
}}
%%
\long\def\startMetapostFrames[#1]#2\stopMetapostFrames{%
\getparameters[mps.] [end=360,start=0,step=1,
      name={out-mps},preamble=preamble,
      mpformat=metafun,#1,body={#2}]
\dostopMetapostFrames
}
%%
\def\dostopMetapostFrames{%
\ctxlua{%
  local preamble =
    MetapostFramesTable[tostring("\csname mps.preamble\endcsname")] or ''
  local end_limit = tonumber(\csname mps.end\endcsname)
  local step_value = tonumber(\csname mps.step\endcsname)
  local start = tonumber(\csname mps.start\endcsname)
  local outdir = tostring("\csname mps.name\endcsname")
  local outfile = tostring("\csname mps.name\endcsname") .. ".mp"
  local mpformat = tostring("\csname mps.mpformat\endcsname")
  local outputtemplate = string.format('outputtemplate := "%s/%s.cmps";',outdir)

```

```

local frames_limit = string.format("numeric frame_min, frame_max; frame_min=%0.5f;
                                   frame_max=%05f;", start, end_limit)
local mpgraphic_body = tostring("\csname mps.body\endcsname")
local start_frame = string.format("for frame:=\%d step \%d until \%d:beginfig(frame);",
                                   start, step_value, end_limit)

local stop_frame = " endfig;endfor;"
local mpgraph = preamble ..
                outputtemplate ..
                frames_limit ..
                start_frame .. mpgraphic_body .. stop_frame ..
                'end.'
dir.mkdirs(outdir)
io.savedata(outfile, mpgraph)
mpost_execute = string.format("mpost --mem=\%s \%s ", mpformat, outfile)
os.execute(mpost_execute)
}}

```

Note: I assume that `metafun` and `METAPOST` formats are placed in the same directory as the animation file and that I'm saving all `mps` (i.e. `postscript`) files in the local folder where the animation file resides.

The next step is to create a single `pdf` from all these `*.mps` and then a `swf`, but these two steps can be merged into a `\MakeSwfFromMps` macro that converts all `*.mps` files into a single `swf` file. Here I use the `ConTeXt-mkiv` snippet

```

\starttext
\pagefigure[folder/fig1.mps]
\pagefigure[folder/fig2.mps]
\stoptext

```

to convert `folder/fig1.mps` and `folder/fig2.mps` into one `pdf`, and then I call `pdf2swf` on the resulting `pdf`.

```

\def\MakeSwfFromMps[#1]{%
\getparameters[mkpdfmps.] [end=360, start=0, step=1, name={out-mps}, fps=60, #1]
\ctxlua{%
local outdir=tostring("\csname mkpdfmps.name\endcsname")
local outfile_tex=tostring("\csname mkpdfmps.name\endcsname") .. ".tex"
local outfile_pdf=tostring("\csname mkpdfmps.name\endcsname") .. ".pdf"

```

```

local end_limit=tonumber(\csname mkpdfmps.end\endcsname)
local start = tonumber(\csname mkpdfmps.start\endcsname)
local step = tonumber(\csname mkpdfmps.step\endcsname)
local fps = tonumber(\csname mkpdfmps.fps\endcsname)
local tex_body = ''
local tex_snippet
local context_execute
local pdf2swf_execute
for i=start,end_limit,step do
  tex_body = tex_body .. string.format("\pagefigure[\%s/\%06d]",outdir, i)
end
tex_snippet =
  "\nopdfcompression\pdfminorversion4\starttext " .. tex_body .. "\stoptext"
io.savedata(outfile_tex,tex_snippet)
context_execute = string.format("context --batchmode \%s" ,outfile_tex)
os.execute(context_execute)
pdf2swf_execute = string.format("pdf2swf --quiet -s framerate=\%s \%s" ,fps ,outfile_pdf)
os.execute(pdf2swf_execute)
}}
\protect

```

4 Example

I attempt to imitate <http://melusine.eu.org/syracuse/metapost/animations/chupin/?idsec=filtre> by creating four animations of the function tran that share the same transfert preamble:

```

\startPreambleMetapostFrames[name=transfert]
path Hcurve;
% Fonction de calcul du module du filtre
vardef trans(suffix pole,zero)(expr nr_poles,nr_zeros,w) =
  save H;
  numeric H;
  H:=1;
  for i:=0 upto (nr_poles-1): H:=H/abs(w-pole[i]); endfor;
  for i:=0 upto (nr_zeros-1): H:=H*abs(w-zero[i]); endfor;
  H
enddef;

```

```

def doit(suffix pole,zero)(expr nr_poles,nr_zeros,frame,u,max_estimated) =
  %%setbounds currentpicture to fullcircle scaled (2u+.5u);
  numeric i;
  pair p[];
  pair w ;

  i:=frame/2;
  w := (cosd(i)*u,sind(i)*u);
  H:=trans(pole,zero)(nr_poles,nr_zeroes,w);
  if i=0:
    Hcurve := (0,H*u);
  else:
    Hcurve := Hcurve--(i,H*u);
  fi;
  draw Hcurve withcolor red withpen pencircle scaled 2pt;
  drawarrow (frame_min,0) -- (frame_max/2+0.1*frame_max,0);
  drawarrow (frame_min,0) -- (0,max_estimated); %% estimated

  picture p ;
  p := image (
    draw fullcircle scaled 2u withcolor 0.8white withpen pencircle scaled 0.5pt;
    draw (0,0) -- (cosd(i)*u,sind(i)*u) withcolor black withpen pencircle scaled 0.5pt;
    drawarrow (-1.1u,0) -- (1.2u,0) withcolor 0.8white withpen pencircle scaled 0.5pt;
    drawarrow (0,-1.1u) -- (0,1.2u) withcolor 0.8white withpen pencircle scaled 0.5pt;
    for k:=0 upto (nr_zeroes-1):
      drawdot zero[k] withcolor blue withpen pencircle scaled 2pt ;
    endfor;
    for k:=0 upto (nr_poles-1):
      drawdot pole[k] withcolor red withpen pencircle scaled 2pt ;
    endfor;
  );
  draw p shifted (-1.3u,0);

enddef;
\stopPreambleMetapostFrames
%%
%%
%%
%%

```

```

%% Main code
\starttext
\startMetapostFrames[name=trasf1,preamble=transfert]
  numeric u;
  u:=1cm;
  numeric max_estimated ;
  max_estimated := 7*u;

  % zeros and poles
  pair zero[],pole[];
  numeric nr_zeroes, nr_poles;
  nr_zeroes:=2;
  nr_poles:=2;
  zero[0]:=(0.86u,0.5u);
  zero[1]:=(-.42u,0.90u);
  pole[0]:=(-0.52u,0.26u);
  pole[1]:=(0.55u,0.58u);
  doit(pole,zero)(nr_poles,nr_zeros,frame,u,max_estimated) ;
\stopMetapostFrames
\MakeSwfFromMps[name={trasf1}]
%
\startMetapostFrames[name=trasf2,preamble=transfert]
  numeric u;
  u:=1cm;
  numeric max_estimated ;
  max_estimated := 7*u;

  % zeros and poles
  pair zero[],pole[];
  numeric nr_zeroes, nr_poles;
  nr_zeroes:=2;
  nr_poles:=2;
  zero[0]:=(0.86u,0.5u);
  zero[1]:=(-.42u,0.90u);
  pole[0]:=(-0.52u,0.26u);
  pole[1]:=(0.25u,0.29u);
  doit(pole,zero)(nr_poles,nr_zeros,frame,u,max_estimated) ;
\stopMetapostFrames
\MakeSwfFromMps[name={trasf2}]
%
```

```

\startMetapostFrames[name=trasf3,preamble=transfert]
  numeric u;
  u:=1cm;
  numeric max_estimated ;
  max_estimated := 7*u;

  % zeros and poles
  pair zero[],pole[];
  numeric nr_zeroes, nr_poles;
  nr_zeroes:=2;
  nr_poles:=2;
  zero[0]:=(0.86u,0.5u);
  zero[1]:=(-.42u,0.90u);
  pole[0]:=(-0.52u,0.26u);
  pole[1]:=(0.1u,0.1u);
  doit(pole,zero)(nr_poles,nr_zeros,frame,u,max_estimated) ;
\stopMetapostFrames
\MakeSwfFromMps[name={trasf3}]

\startMetapostFrames[name=trasf4, preamble=transfert]
  numeric u;
  u:=1cm;
  numeric max_estimated ;
  max_estimated := 7*u;

  % zeros and poles
  pair zero[],pole[];
  numeric nr_zeroes, nr_poles;
  nr_zeroes:=2;
  nr_poles:=2;
  zero[0]:=(1u,0);
  zero[1]:=(-1u,0);
  pole[0]:=(cosd(135)*0.8u,sind(135)*0.8u);
  pole[1]:=(cosd(45)*0.7u,sind(45)*0.7u);
  doit(pole,zero)(nr_poles,nr_zeros,frame,u,max_estimated) ;
\stopMetapostFrames
\MakeSwfFromMps[name={trasf4}]
%
\setbox1=\vbox{\flashmoviectx[width=0.45\textwidth,
                           height=0.25\textheight,file=trasf1.swf]}

```

```
\setbox2=\vbox{\flashmoviectx[width=0.45\textwidth,  
                                height=0.25\textheight,file=trasf2.swf]}  
\setbox3=\vbox{\flashmoviectx[width=0.45\textwidth,  
                                height=0.25\textheight,file=trasf3.swf]}  
\setbox4=\vbox{\flashmoviectx[width=0.45\textwidth,  
                                height=0.25\textheight,file=trasf4.swf]}  
  
\hbox{\copy1\copy2}  
\hbox{\copy3\copy4}  
%  
\stoptext
```

(If you opened this file using AdobeReader 9, you should see four animations right here.)

5 Conclusion

This first attempt showed that it is possible to use ConTEXt-mkiv to make interesting animations with Flash, but at this stage `supp-swf.mkiv` must be considered as a "sketch". However, this was enough to get Hans Hagen interested, and he immediately showed me two lua files (`grph-swf.lua` and `lpdf-swf.lua`) that replace `supp-swf.mkiv` in the correct ConTEXt-mkiv way. As a result, I can now research the next steps in the following order:

1. complete `back-swf.mkiv`, `grph-swf.lua` and `lpdf-swf.lua` in a consistent manner (they are already in the `minimals-beta` distribution);
2. study how to use the internal `mplib`, and see if is possible to avoid a call to the external context process;
3. provide the end user with the parameters used by `pdf2swf` for the conversion from pdf to swf;
4. study a Lua binding for `pdf2swf` — there is already a `gfx.so` shared library, but it used for the Python binding.
5. verify the performance of AdobeReader and test the limits of `pdf2swf`.