MetaPost

Extensions

A few examples

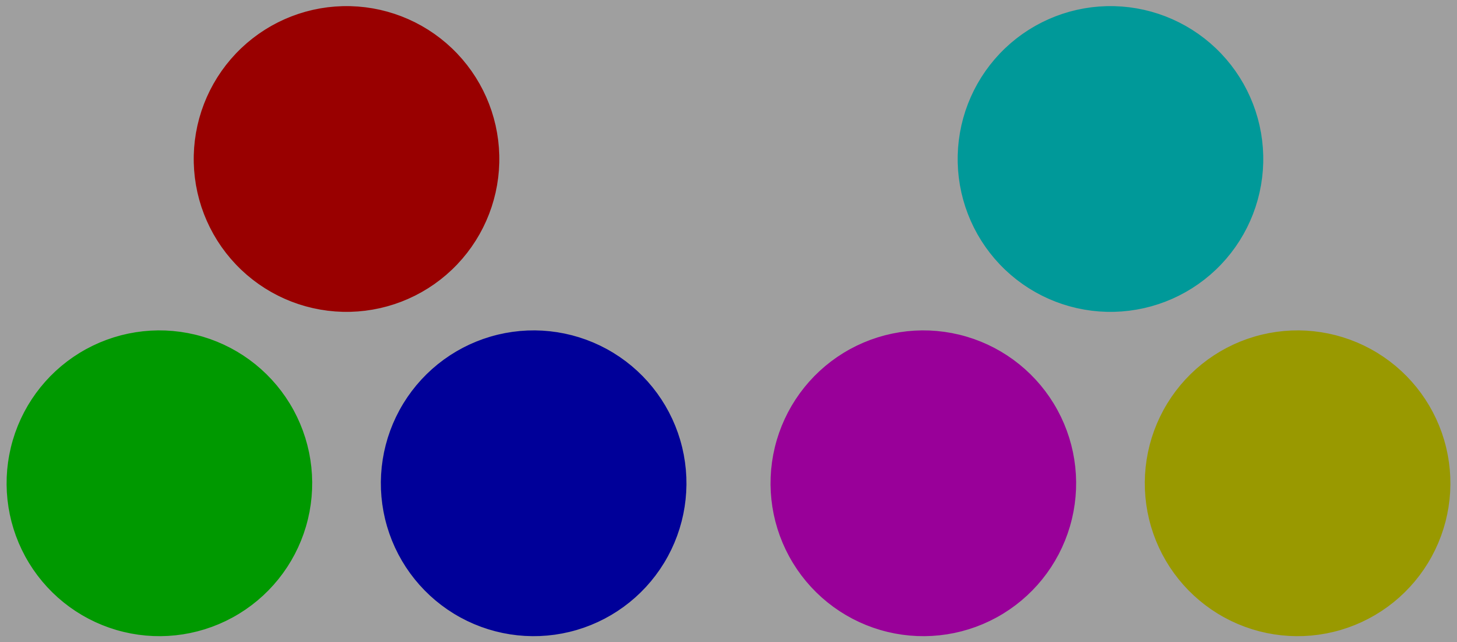
BachoTEX 2018 — Hans Hagen

History

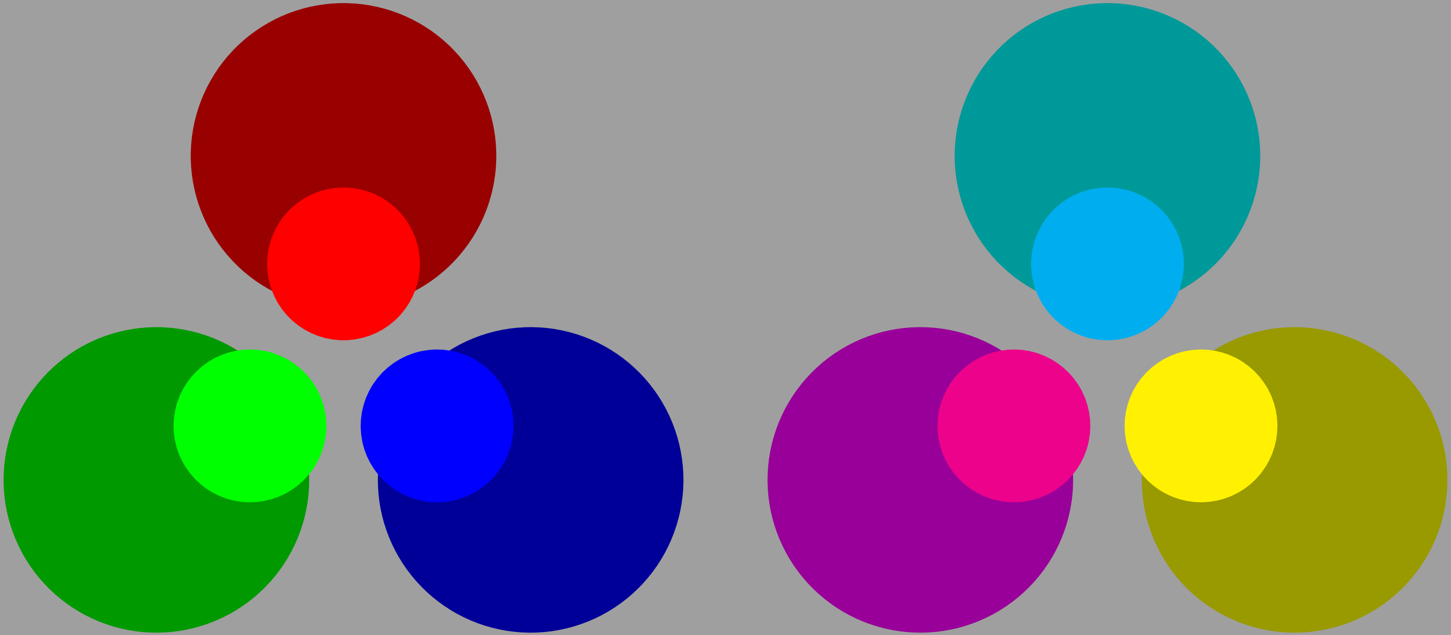
- We started using MetaPost some two decades ago and immediately went the pdf route.
- We used special colors plus specials to communicate extensions, for instance cmyk colors and shades.
- This mechanism was stepwise improved and extended. Some mechanisms, like texts, needed an extra pass.
- When we moved to LuaTEX and mplib we started using pre- and postscripts to carry information with the paths.
- Currently we use a bit of Lua from within mplib to communicate during the MetaPost run with ConTEXt. This permits cleaner interfaces.

Colors

```
\startMPcode
  draw image (
    draw image (
      fill unitcircle rotated  45 withcolor "red" ;
      fill unitcircle rotated 165 withcolor "green" ;
      fill unitcircle rotated 285 withcolor "blue" ;
    ) shifted (-1.25,0)  ;
    draw image (
      fill unitcircle rotated  45 withcolor "cyan" ;
      fill unitcircle rotated 165 withcolor "magenta" ;
      fill unitcircle rotated 285 withcolor "yellow" ;
    ) shifted ( 1.25,0) ;
  ) xsized TextWidth;
\stopMPcode
```

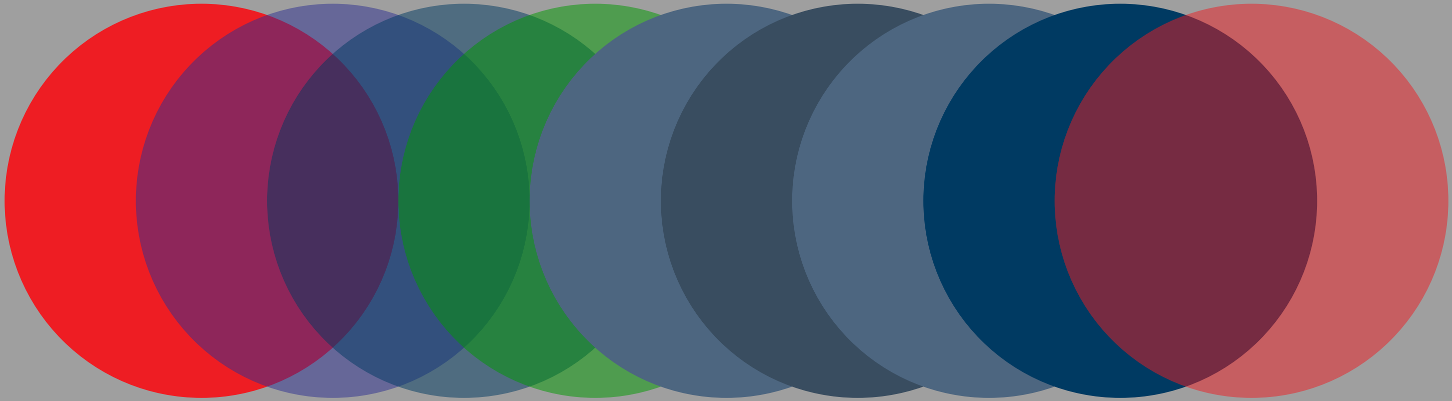


```
\definecolor[red]    [r=.6]
\definecolor[green]  [g=.6]
\definecolor[blue]   [b=.6]

\definecolor[cyan]   [g=.6,b=.6]
\definecolor[magenta][r=.6,b=.6]
\definecolor[yellow] [r=.6,g=.6]

\startMPcode
  draw image (
    draw image (
      fill unitcircle rotated  45 withcolor "red" ;
      fill unitcircle rotated 165 withcolor "green" ;
      fill unitcircle rotated 285 withcolor "blue" ;
      fill unitcircle rotated  45 scaled 2/4 withcolor (1,0,0) ;
      fill unitcircle rotated 165 scaled 2/4 withcolor (0,1,0) ;
      fill unitcircle rotated 285 scaled 2/4 withcolor (0,0,1) ;
    ) shifted (-1.25,0)  ;
    draw image (
      fill unitcircle rotated  45 withcolor "cyan" ;
      fill unitcircle rotated 165 withcolor "magenta" ;
      fill unitcircle rotated 285 withcolor "yellow" ;
      fill unitcircle rotated  45 scaled 2/4 withcolor (1,0,0,0) ;
      fill unitcircle rotated 165 scaled 2/4 withcolor (0,1,0,0) ;
      fill unitcircle rotated 285 scaled 2/4 withcolor (0,0,1,0) ;
    ) shifted ( 1.25,0) ;
  ) xsized TextWidth;
\stopMPcode
```



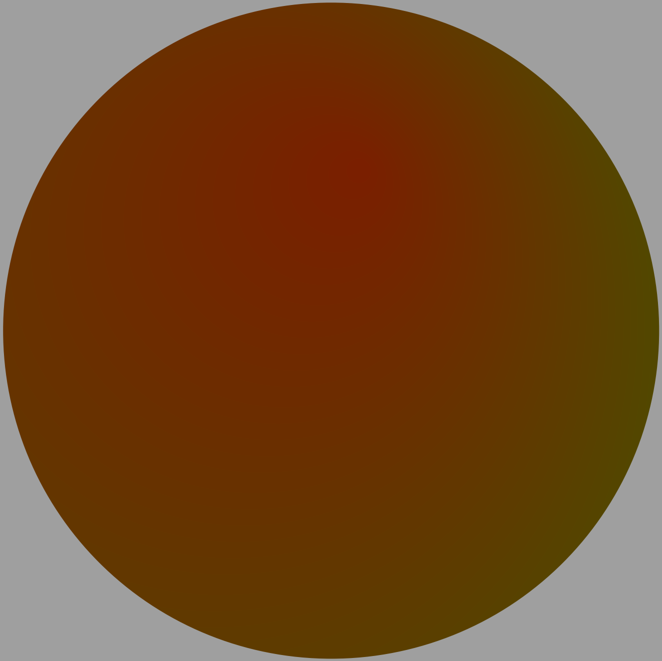
```
\definecolor     [whatever]   [c=1,a=1,t=0.5]

\definecolor     [blue]       [c=1,m=.38,y=0,k=.64] % pantone pms 2965
  uncoated m
\definecolor     [yellow]     [c=0,m=.28,y=1,k=.06] % pantone pms  124
  uncoated m

\definespotcolor [blue-100]   [blue]   [p=1]
\definespotcolor [yellow-100] [yellow] [p=1]

\definemultitonecolor [somecolor] [blue=.12,yellow=.28] [c=.1,m=.1,y=.3,k=.1
  ]
```

```
% \enabletrackers[metapost.lua]
\startMPcode
vardef C(expr r,dx) = fullcircle scaled r shifted (dx,0) enddef ;

draw image (
  fill C(3cm,1cm) withcolor (0,1,1,0) ;
  fill C(3cm,2cm) withcolor transparent(1,0.5,(1,1,0,0)) ;
  fill C(3cm,3cm) withcolor transparent(1,0.5,"blue-100") ;
  fill C(3cm,4cm) withcolor 0.75*transparent(1,0.5,"green") ;
  fill C(3cm,5cm) withcolor spotcolor("blue-100",(.3,.4,.5)) ;
  fill C(3cm,6cm) withcolor 0.75 * spotcolor("blue-100",(.3,.4,.5)) ;
  fill C(3cm,7cm) withcolor namedcolor("blue-100") ;
  fill C(3cm,8cm) withcolor "blue-100" ;
  fill C(3cm,9cm) withcolor (0,1,1,0) withtransparency (1,0.5);
) xsized TextWidth;
\stopMPcode
```



Shades

```
\startMPcode
  draw image (
    fill fullcircle scaled 10cm
      withshademethod "circular"
      withshadevector (5cm,1cm)
      withshadecenter (.1,.5)
      withshadedomain (.2,.6)
      withshadefactor 1.2
      withshadecolors ("red","green")
    ;
    fill fullcircle scaled 10cm shifted (12cm,0)
      withshademethod "circular"
      withshadevector (4cm,2cm)
      withshadecenter (.2,.8)
      withshadedomain (.2,.8)
      withshadefactor 1.5
      withshadecolors ("blue","green")
    ;
  ) xsized TextWidth ;
\stopMPcode
```



domain	The range over which the colors run, with a minimum of 0 and maximum of 1.
color	A color to start from and one to end with, we default from black to white.
type	The shading can be linear or circular.
center	The origin of the shade vector.
radius	The radius vector of a circular shade.
vector	Where we start and end the shading.

For a **linear** shade the centers are the lower left and upper right corners, for a circular shade it's the center of the path. For a **circular** shade the radius runs from zero to the maximum distance from the center as determined by the boundingbox.

The vector is used as follows: the first coordinate (xpart) determines the point on the path where we start, the second coordinate (ypart) the point on the path where we end.

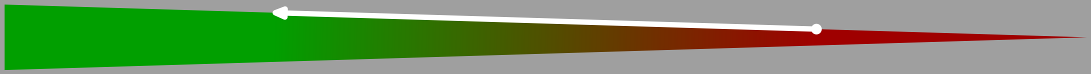
```
\startreusableMPgraphic{bullet}
   fill fullcircle
     scaled (.75EmWidth)
     withshademethod "circular"
     withcolor "red" shadedinto "blue" ;
\stopreusableMPgraphic
```

```
\definesymbol[1][\hbox{\lower.125ex\hbox{\reuseMPgraphic{bullet}}}]
```

- This is item one!
- This is item two!

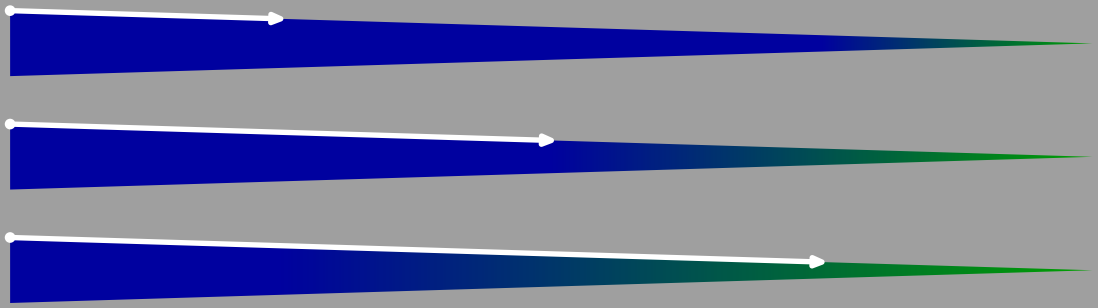
A triangle has three points. Using 1 and 2 as second vector value gives the same results as do values in the range 0 upto 1 and 2 upto 3 (0 again).

```
\startMPcode
fill fulltriangle xyscaled (TextWidth,1cm)
  withshademethod "linear"
  withshadevector (0.25,0.75)
  withshadecolors (darkred,darkgreen)
;

draw fulltriangle xyscaled (TextWidth,1cm)
  shownshadevector (0.25,0.75)
  withpen pencircle scaled 2
  withcolor white ;
\stopMPcode
```



The shadevector relates to (the x coordinates of) points on the path. A variant is to use the boundingbox:

```
\startMPcode
for i=1 upto 3 :
  fill fulltriangle xyscaled (TextWidth,1cm) shifted (0,-i*15mm)
    withshademethod "linear"
    withshadedirection (1,1-i/4)
    withshadecolors (darkgreen,darkblue)
  ;
endfor ;
for i=1 upto 3 :
  draw fulltriangle xyscaled (TextWidth,1cm) shifted (0,-i*15mm)
    shownshadevector (1,1-i/4)
    withpen pencircle scaled 2
    withcolor white ;
endfor ;
\stopMPcode
```

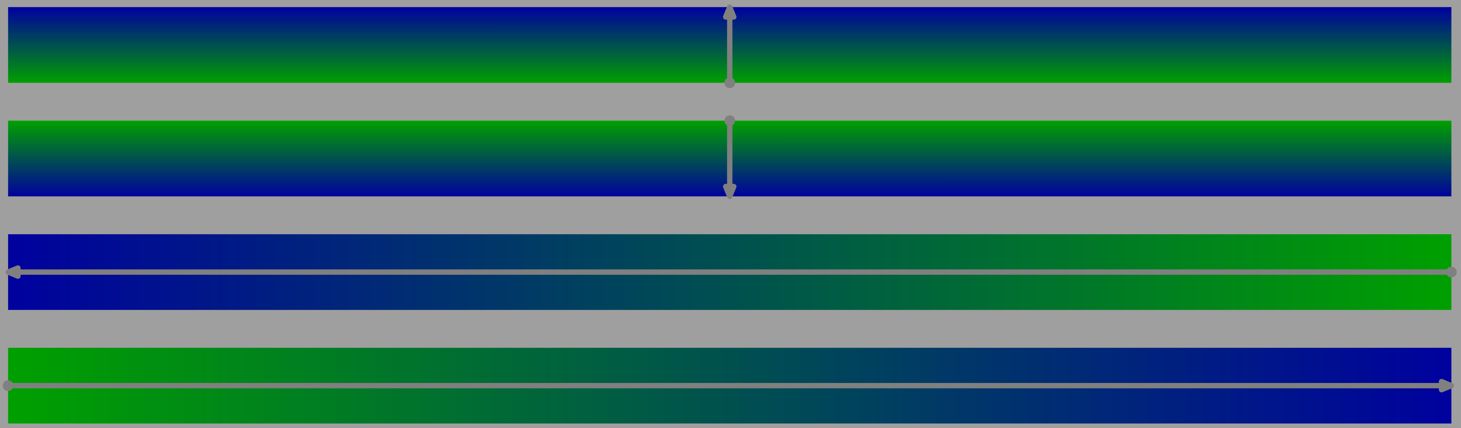


To make life convenient we provide a few constants that indicate directions:

```
\startMPcode
pair shadedup    ; shadedup    := (0.5,2.5) ;
pair shadeddown  ; shadeddown  := (2.5,0.5) ;
pair shadedleft  ; shadedleft  := (1.5,3.5) ;
pair shadedright ; shadedright := (3.5,1.5) ;
\stopMPcode
```

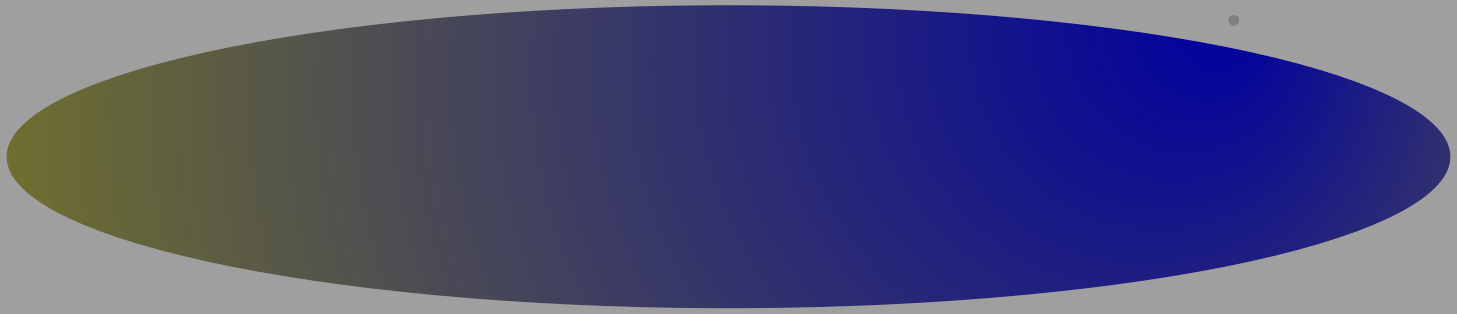
```
\startMPcode
for d = shadedup, shadeddown, shadedleft, shadedright :
  fill fullsquare xyscaled (TextWidth,1cm)
    withshademethod "linear"
    withshadedirection d
    withshadecolors (darkgreen,darkblue)
  ;
  currentpicture := currentpicture shifted (0,15mm) ;
endfor ;

currentpicture := currentpicture shifted (0,-60mm) ;

for d = shadedup, shadeddown, shadedleft, shadedright :
  draw fullsquare xyscaled (TextWidth,1cm)
    shownshadedirection d
    withpen pencircle scaled 2
    withcolor .5white ;
  currentpicture := currentpicture shifted (0,15mm) ;
endfor ;
```

```
\stopMPcode
```



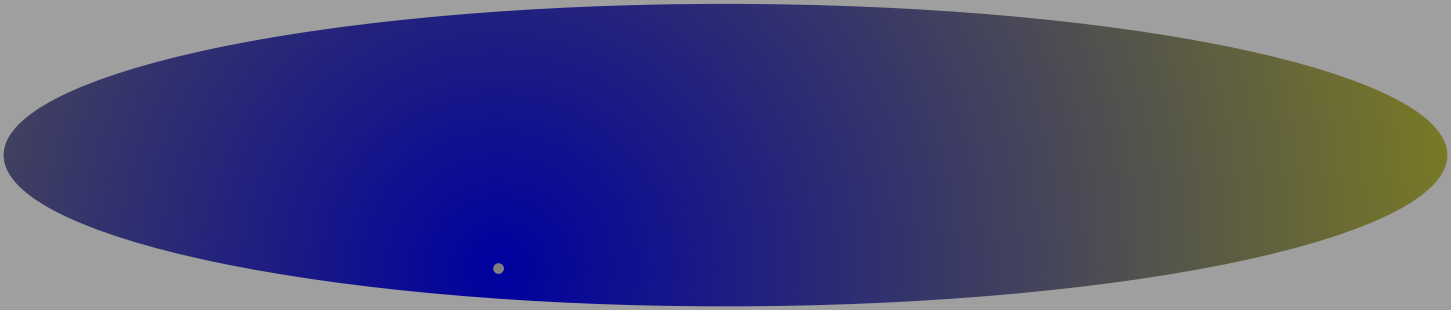
In case of a circular shade another method comes in handy. Here the values relate to the center of path i.e. they shift the center by the given fraction of the width and height of the boundingbox devided by 2.

```
\startMPcode
fill fullcircle xyscaled (TextWidth,4cm)
  withshademethod "circular"
  withshadecenter (.7,.9)
  withshadecolors (darkblue,darkyellow)
;

draw fullcircle xyscaled (TextWidth,4cm)
  shownshadecenter (.7,.9)
  withpen pencircle scaled 2
  withcolor .5white ;
\stopMPcode
```



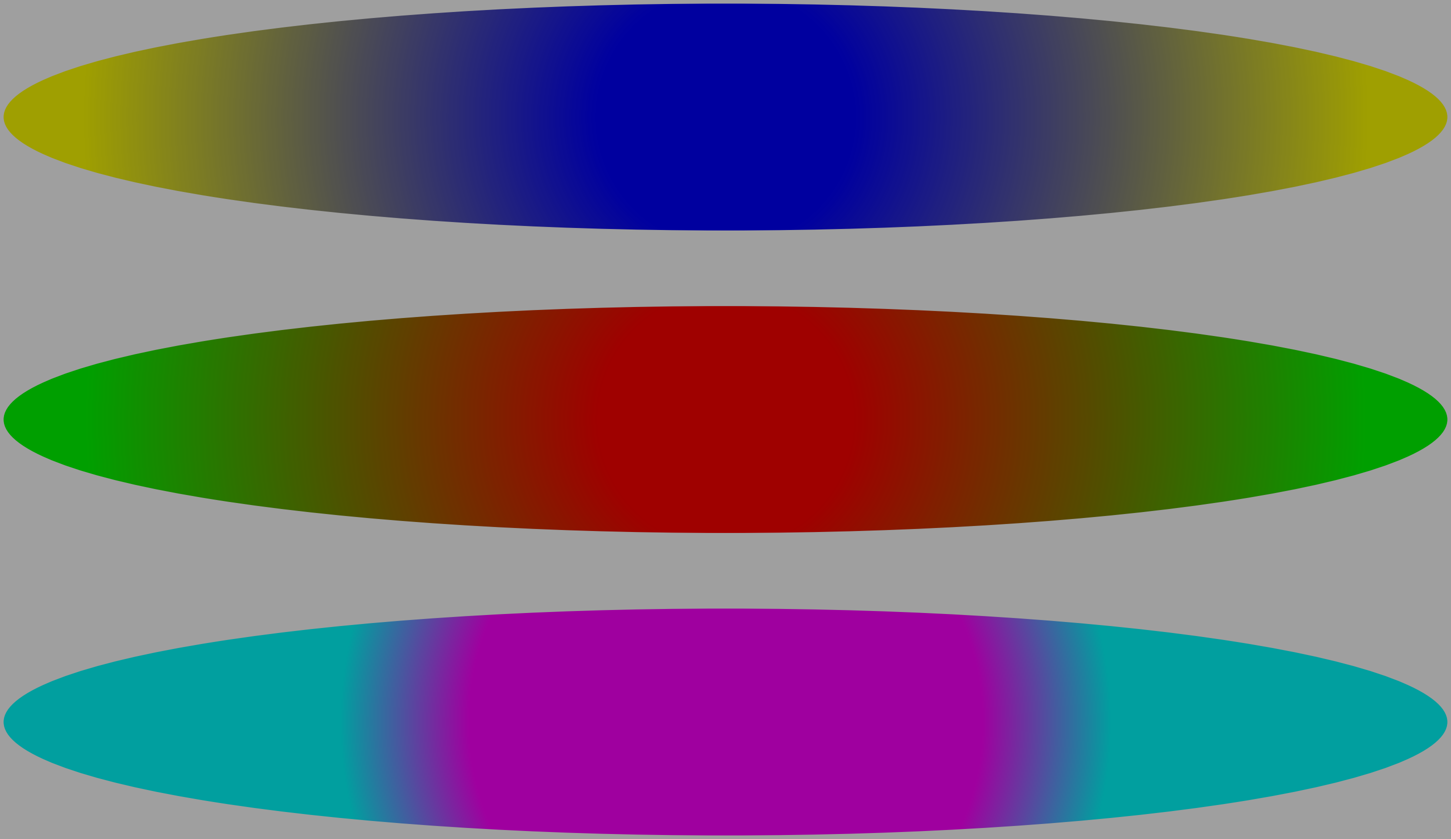
You can set a center directly i.e. unrelated to the center of the path as follows:

```
\startMPcode
fill fullcircle xyscaled (TextWidth,4cm)
  withshademethod "circular"
  withshadeorigin (-30mm,-15mm)
  withshadecolors (darkblue,darkyellow)
;

draw fullcircle xyscaled (TextWidth,4cm)
  shownshadeorigin (-30mm,-15mm)
  withpen pencircle scaled 2
  withcolor .5white ;
\stopMPcode
```



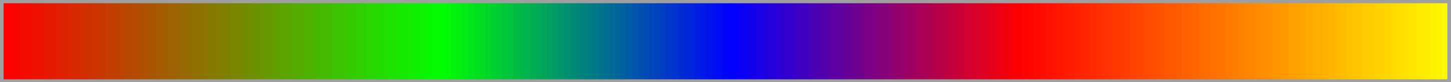
In a similar way you can set an explicit radius:

```
\startMPcode
fill fullcircle xyscaled (TextWidth,3cm)
  withshademethod "circular"
  withshaderadius (10mm,50mm)
  withshadecolors (darkblue,darkyellow)
;

currentpicture := currentpicture shifted (0,40mm) ;

fill fullcircle xyscaled (TextWidth,3cm)
  withshademethod "circular"
  withshaderadius (50mm,10mm)
  withshadecolors (darkgreen,darkred)
;

currentpicture := currentpicture shifted (0,40mm) ;

fill fullcircle xyscaled (TextWidth,3cm)
  withshademethod "circular"
  withshaderadius (20mm,30mm)
  withshadecolors (darkmagenta,darkcyan)
;
\stopMPcode
```



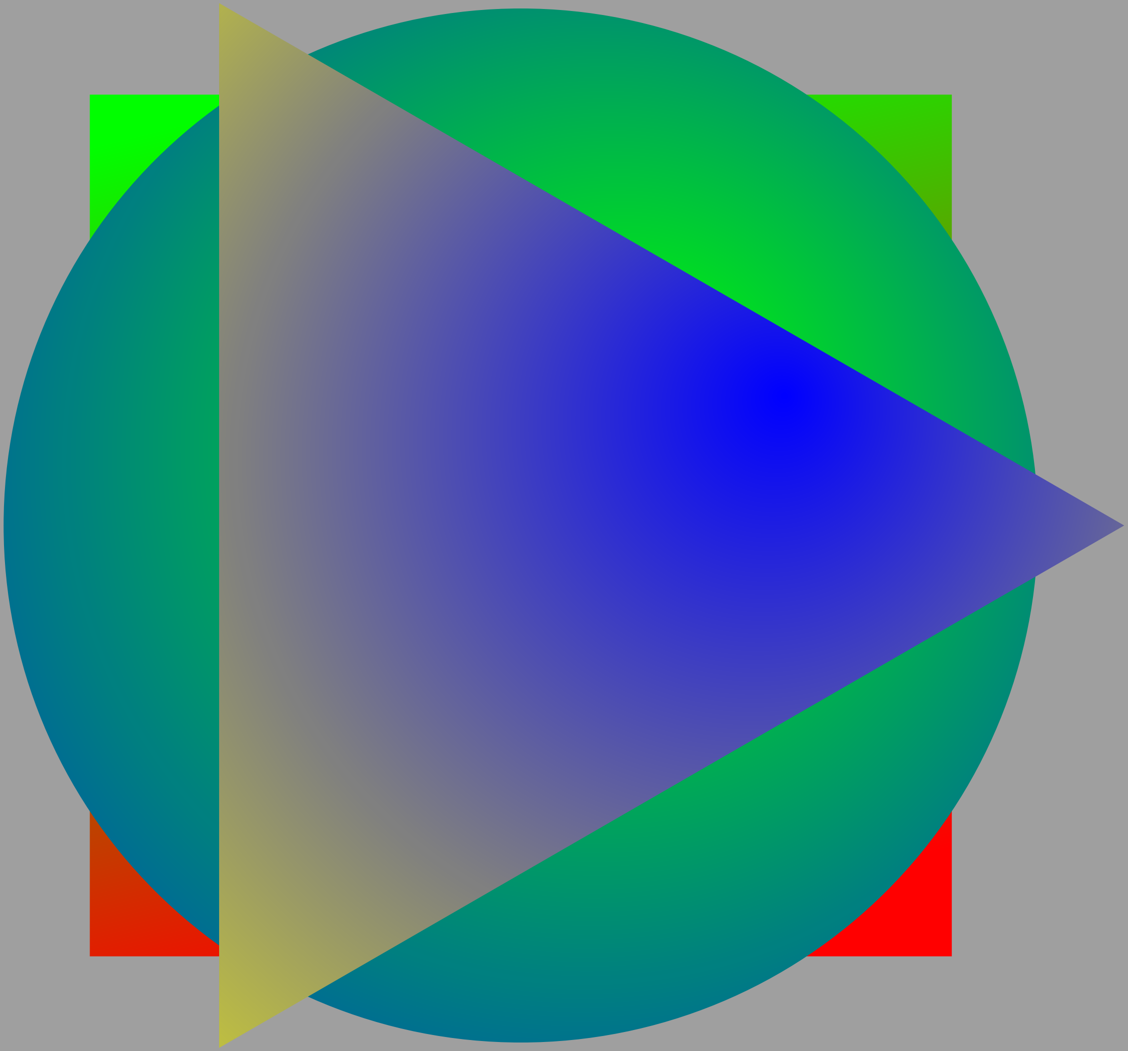
This one is made for Mojca:

```
\startMPcode
fill fullsquare xyscaled (TextWidth,1cm)
  withshademethod "linear"
  withshadevector (0,1)
  withshadestep (
    withshadefraction .3
    withshadecolors (red,green)
  )
  withshadestep (
    withshadefraction .5
    withshadecolors (green,blue)
  )
  withshadestep (
    withshadefraction .7
    withshadecolors (blue,red)
  )
  withshadestep (
    withshadefraction 1
    withshadecolors (red,yellow)
  )
;
\stopMPcode
```

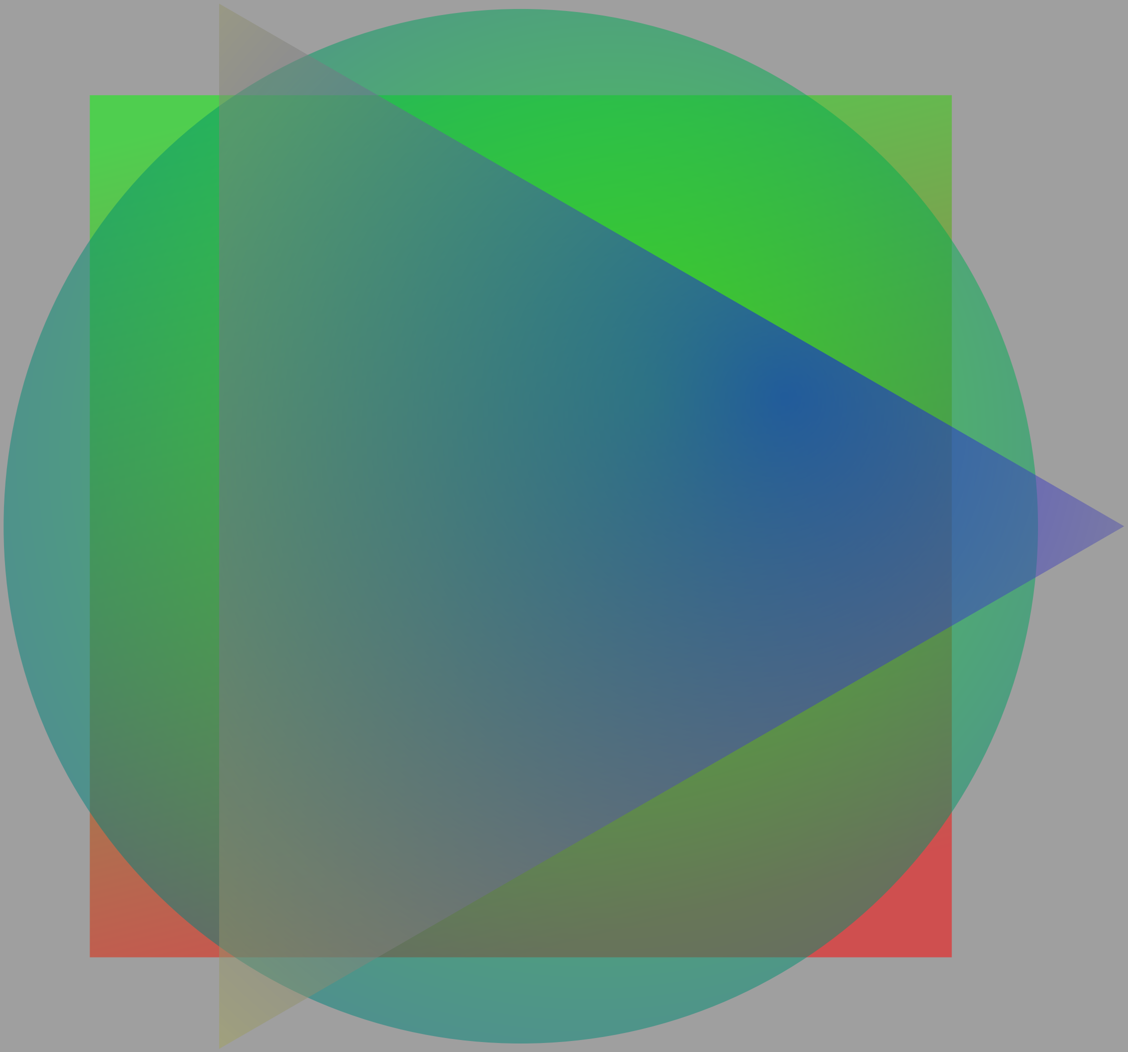


Shades work well with colors and transparencies. This involves quite some resource managament in the backend but it’s hidden by the interface.

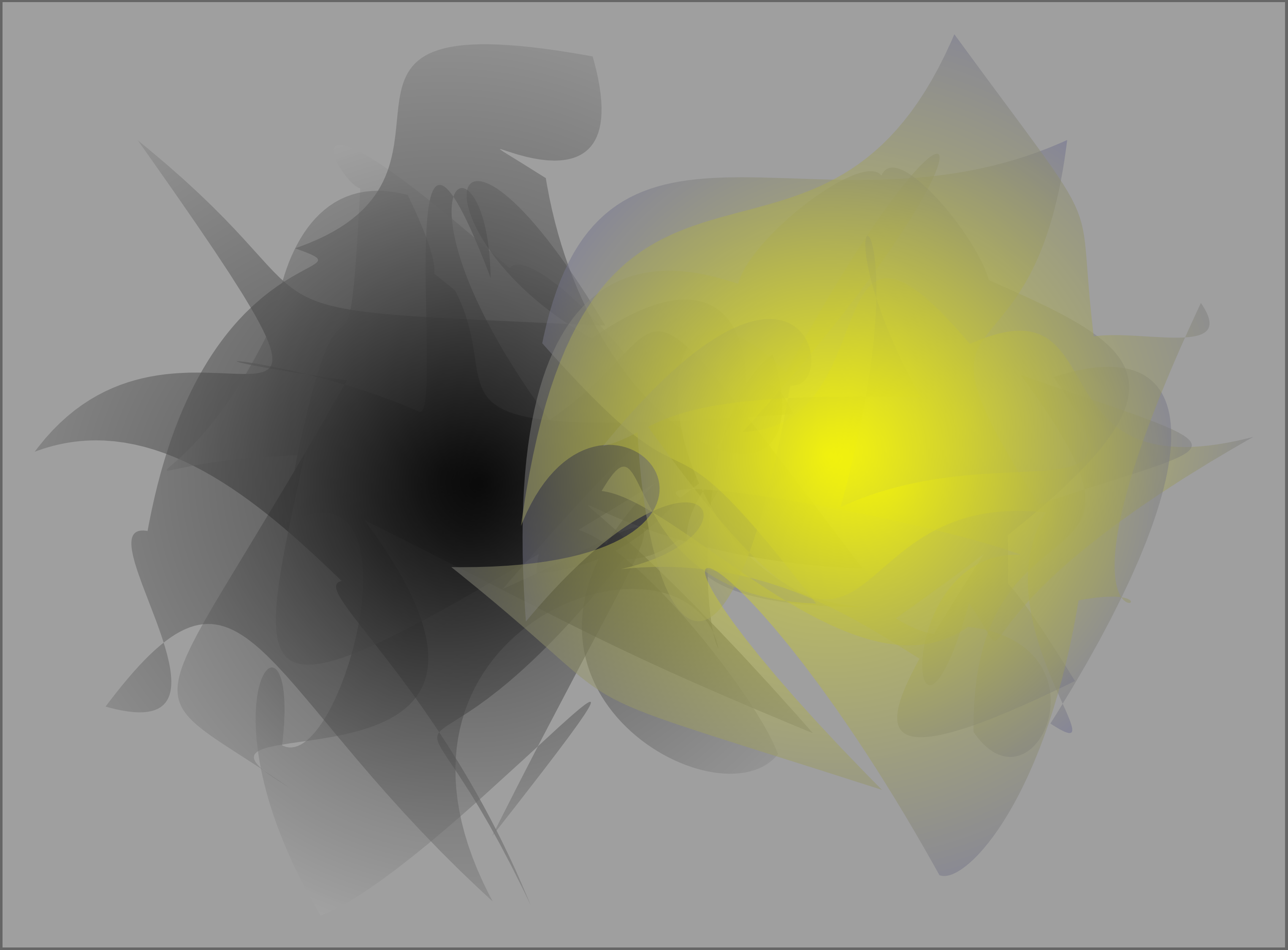
```
\startMPcode
draw image (
  fill fullsquare scaled 5cm
    withshademethod "linear"
    withshadefactor 1
    withshadedomain (0,1)
    withshadevector (0.5,2.75)
    withshadecolors (red,green) ;

  fill fullcircle scaled 6cm
    withshademethod "circular"
    withshadefactor 1
    withshadedomain (0,1)
    withshadecenter (.25,.25)
    withshadecolors (green,blue) ;

  fill fulltriangle scaled 7cm
    withshademethod "circular"
    withshadefactor 1
    withshadedomain (0,1)
    withshadecenter (.25,.25)
    withshadecolors (blue,yellow) ;
) ysized TextHeight ;
\stopMPcode
```



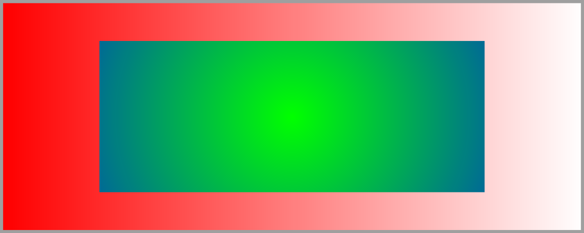
```
\startMPcode
draw image (
  fill fullsquare scaled 5cm
    withshademethod "linear"
    withshadevector (0.5,2.75)
    withshadecolors (red,green)
    withtransparency (1,.5) ;

  fill fullcircle scaled 6cm
    withshademethod "circular"
    withshadecenter (.25,.25)
    withshadecolors (green,blue)
    withtransparency (1,.5) ;

  fill fulltriangle scaled 7cm
    withshademethod "circular"
    withshadecenter (.25,.25)
    withcolor blue shadedinto yellow
    withtransparency (1,.5) ;
) ysized TextHeight ;
\stopMPcode
```



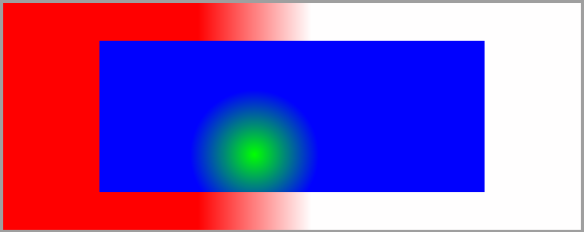
```
\startMPcode
defineshade myshade
  withshademethod "circular"
  withshadefactor 1
  withshadedomain (0,1)
  withshadecolors (black,white)
  withtransparency (1,.5)
;

draw image (
  for i=1 upto 5 :
    fill fullcircle randomized 1 xyscaled(5cm,3cm)
      shaded myshade ;
  endfor ;

  draw image (
    for i=1 upto 5 :
      fill fullcircle randomized 1
        shaded myshade
        withshadecolors (yellow,blue) ;
    endfor ;
  ) xyscaled(5cm,3cm) shifted (5cm,0) ;
) xysized (TextWidth, TextHeight) ;
\stopMPcode
```



```
\startMPcode
fill fullsquare xyscaled (15mm, 15mm)
  withshademethod "linear"
  withshadedirection shadedright
  withshadecolors (red,(1,1,1)) ;

fill fullsquare xyscaled (10mm, 10mm)
  withshademethod "circular"
  withshadecolors (green,blue) ;

currentpicture := currentpicture xysized (.4TextWidth,30mm) ;
currentpicture := currentpicture shifted (5mm,5mm) ;
\stopMPcode
```

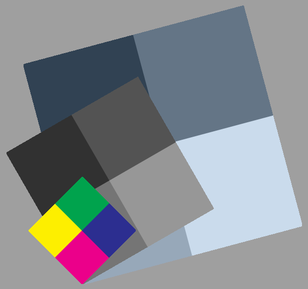


```
\startMPcode
fill fullsquare xyscaled (15mm, 15mm)
  withshademethod "linear"
  withshadetransform "no"
  withshadedirection shadedright
  withshadecolors (red,(1,1,1)) ;

fill fullsquare xyscaled (10mm, 10mm)
  withshademethod "circular"
  withshadetransform "no"
  withshadecolors (green,blue) ;

currentpicture := currentpicture xysized (.4TextWidth,30mm) ;
currentpicture := currentpicture shifted (5mm,5mm) ;
\stopMPcode
```

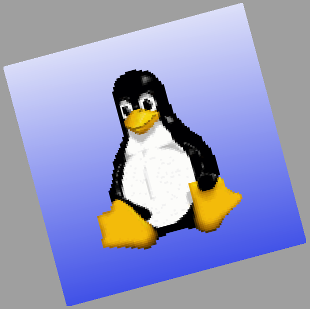


Bitmaps

```
\startMPcode
  draw
    bitmapimage(2,2,"334455 667788 99aabb ccddee")
    scaled 3cm
    rotated 15 ;
  draw
    bitmapimage(2,2,"33 55 77 99")
    scaled 2cm
    rotated 30 ;
  draw
    bitmapimage(2,2,"0000ff00 ff00ff00 00ff0000 ffff0000")
    scaled 1cm
    rotated 45 ;
\stopMPcode
```

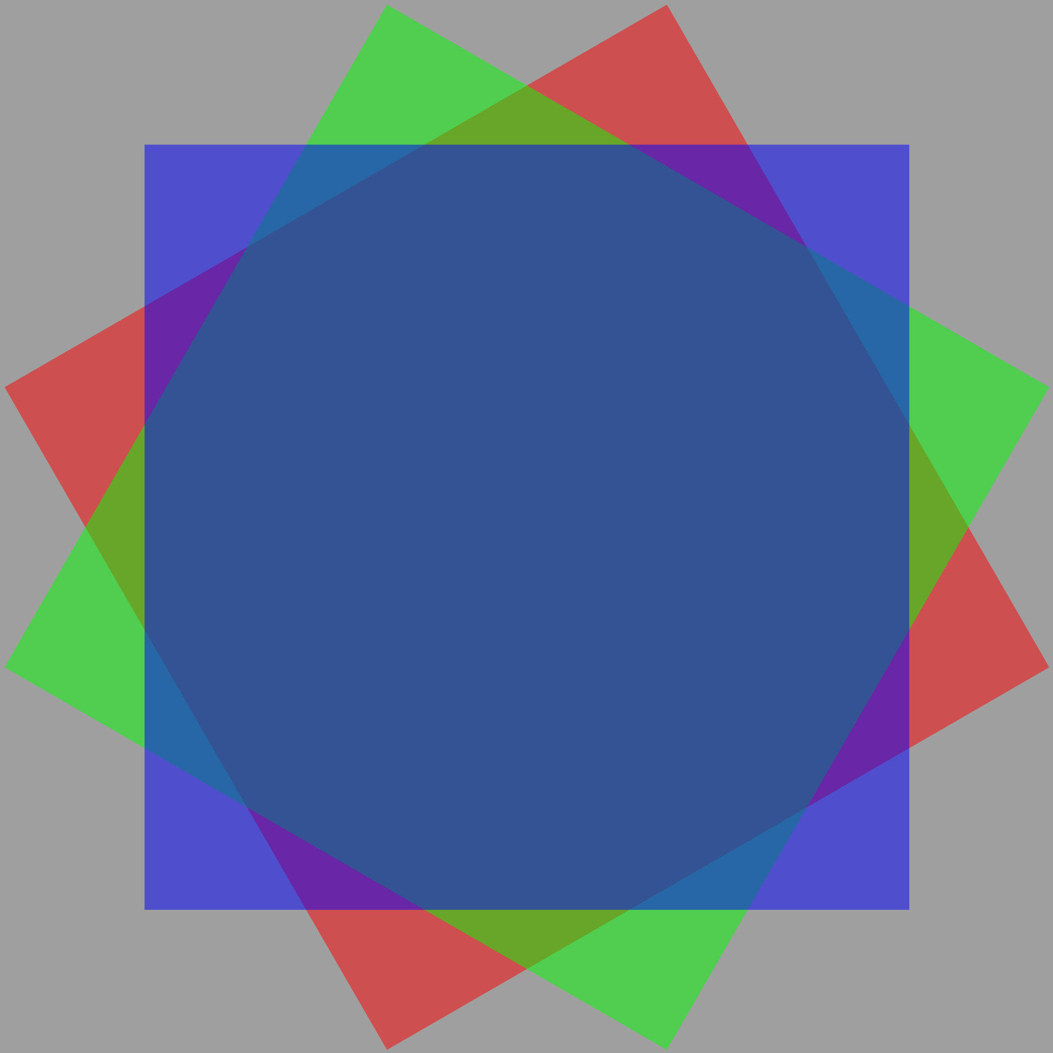


```
\startMPcode
draw bitmapimage (
  128, 128,
  "
      dbdefadbdffbdbdffbdbdffbdbdffbdbdff.....
      dcdffbdcdffbdcdffbdcdffbdcdffbdcdff.....
      dcdffbdcdffbdcdffbdcdffbdcdffbdcdff.....
      ......
  "
) rotated 15 ysized 4cm ;
\stopMPcode
```



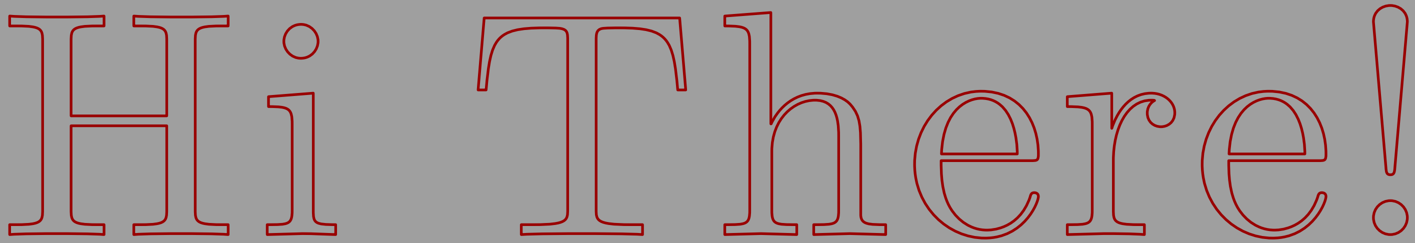
Layers

```
\defineviewerlayer[rotation:30]
\defineviewerlayer[rotation:60]
\defineviewerlayer[rotation:90]

\startMPcode
draw image (

 fill fullsquare scaled 8cm rotated 30
   withcolor red
   withtransparency(1,.5)
   onlayer "rotation:30" ;

 fill fullsquare scaled 8cm rotated 60
   withcolor green
   withtransparency(1,.5)
   onlayer "rotation:60" ;

 fill fullsquare scaled 8cm rotated 90
   withcolor blue
   withtransparency(1,.5)
   onlayer "rotation:90" ;

) ysized TextHeight ;
\stopMPcode
```



Outlines

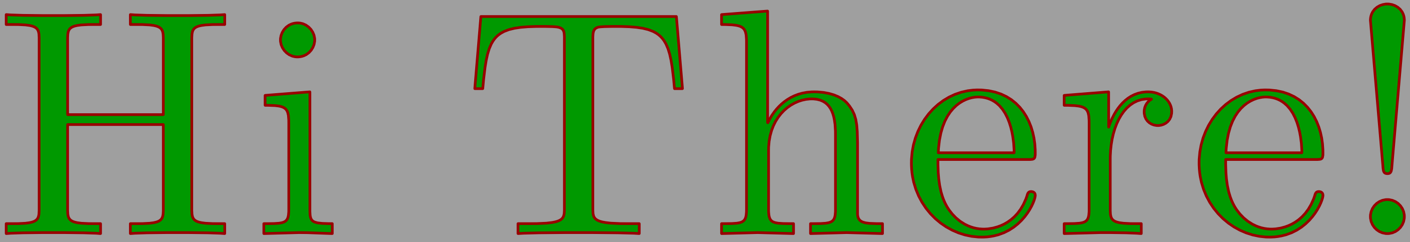
```
\startMPcode
  draw outlinetext.d
    ("Hi There!")
    (withcolor "red" withpen pencircle scaled 1/10 )
    scaled 10 ;
\stopMPcode
```



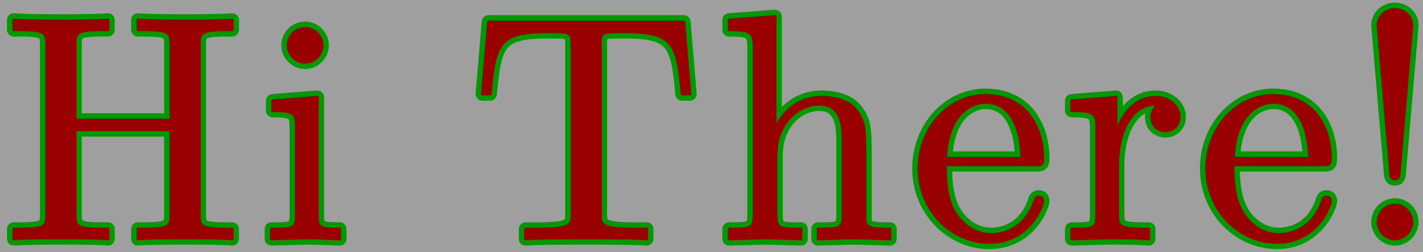
```
\startMPcode
  draw outlinetext.f
   ("Hi There!")
   (withcolor "green")
   scaled 10 ;
\stopMPcode
```

Hi There!

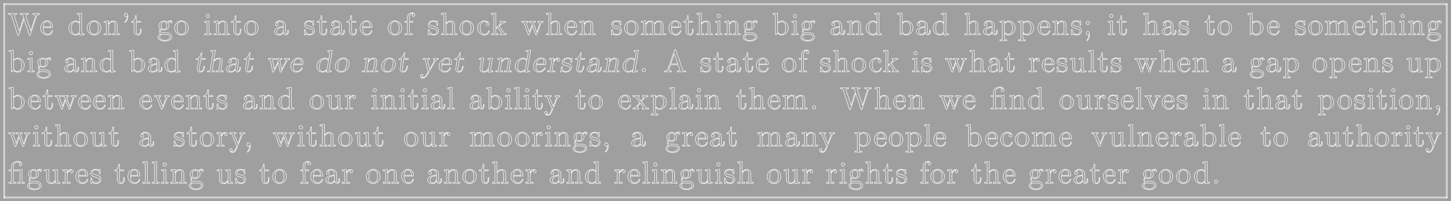
```
\startMPcode
  draw outlinetext.b
    ("Hi There!")
    (withcolor "green")
    (withcolor "red" withpen pencircle scaled 1/10 )
    scaled 10 ;
\stopMPcode
```



```
\startMPcode
  draw outlinetext.r
    ("Hi There!")
    (withcolor "green")
    (withcolor "red" withpen pencircle scaled 1/10 )
    scaled 10 ;
\stopMPcode
```



```
\startMPcode
  draw outlinetext.d
    ("\framed[align=normal]{\input klein }")
    (withcolor "white" withpen pencircle scaled 1/10 )
    xsized TextWidth ;
\stopMPcode
```

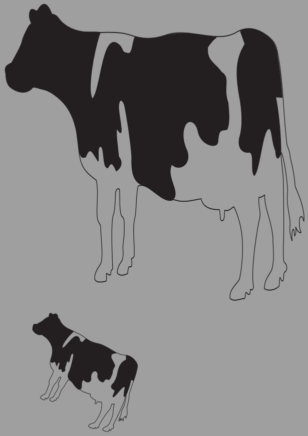


We don’t go into a state of shock when something big and bad happens; it has to be something big and bad *that we do not yet understand*. A state of shock is what results when a gap opens up between events and our initial ability to explain them. When we find ourselves in that position, without a story, without our moorings, a great many people become vulnerable to authority figures telling us to fear one another and relinguish our rights for the greater good.

Images

```
\startMPcode
  draw externalfigure ("cow.pdf") xsized 4cm ;
\stopMPcode
```

```
\startMPcode
  draw figure ("cow.pdf") rotated -25 xsized 2cm shifted (14cm,-3cm) ;
\stopMPcode
```



Text

```
\startMPcode
draw textext("\bfd Hello, {\green does} this work?")
  shifted (4cm,2cm)
  rotated 10
  withcolor white ;

draw textext("\bfd Hello, {\green does} this work?")
  shifted (4cm,-2cm)
  rotated -10
  withcolor white
  withtransparency (1,0.5);

for i=1 step 10 until 360:
  draw textext(decimal i)
    shifted (0,4.5cm)
    rotated i
    withcolor i/360 ;
endfor ;
\stopMPcode
```

1 11 21 31 41 51 61 71 81 91 101 111 121 131 141 151 161 171 181 191 201 211 221 231 241 251 261 271 281 291 301 311 321 331 341 351

Hello, does this work?

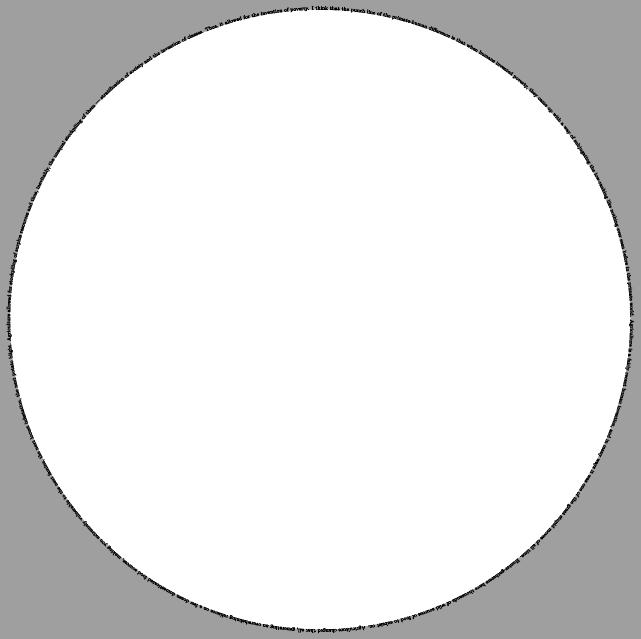
Hello, does this work?

Paths

```
\startMPcode
draw image (
  path p ; p := reverse fullcircle scaled 4cm ;
  draw p ;
  draw followtext(p,
    "A nice clip: Rai Thistlethwayte’s Betty Page @ Keyscape.\quad")
) ysized .6TextHeight ;
\stopMPcode
```

A nice clip: Rai Thistlethwayte's Betty Page @ Keyscape.

```
\startMPcode
draw image (
  path p ; p := fullcircle scaled 4cm ;
  fill p withcolor white ;
  draw followtext(reverse p,"\obeydiscretionaries\samplefile{sapolsky}") ;
) ysized .6TextHeight ;
\stopMPcode
```


Agriculture is a fairly recent human invention, and in many ways it was one of the great stupid moves of all time. Hunter-gatherers have thousands of wild sources of food to subsist on. Agriculture changed that all, generating an overwhelming reliance on a few dozen domesticated food sources, making you extremely vulnerable to the next famine, the next locust infestation, the next potato blight. Agriculture allowed for stockpiling of surplus resources and thus, inevitably, the unequal stockpiling of them — stratification of society and the invention of classes. Thus, it allowed for the invention of poverty. I think that the punch line of the primate-human difference is that when humans invented poverty, they came up with a way of subjugating the low-ranking like nothing ever seen before in the primate world.

Agriculture is a fairly recent human invention, and in many ways it was one of the great stupid moves of all time. Hunter-gatherers have thousands of wild sources of food to subsist on. Agriculture changed that all, generating an overwhelming reliance on a few dozen domesticated food sources, making you extremely vulnerable to the next famine, the next locust infestation, the next potato blight. Agriculture allowed for stockpiling of surplus resources and thus, inevitably, the unequal stockpiling of them — stratification of society and the invention of classes. Thus, it allowed for the invention of poverty. I think that the punch line of the primate-human difference is that when humans invented poverty, they came up with a way of subjugating the low-ranking like nothing ever seen before in the primate world.

So . . .

- Get rid of old code snippets. And maybe translate some experiments into useful code.
- Optimize some of the code. On the average the code is quite efficient but less is often better.
- Check the MetaFun manual for recent additions. And maybe remove older (more MpIIish) solutions.
- Think about a way to circumvent unwanted suffix expansion so that we can use more keywords without problems. (Maybe I should come up with a decent MetaPost extension. Needs discussion with Alan Braslau.)