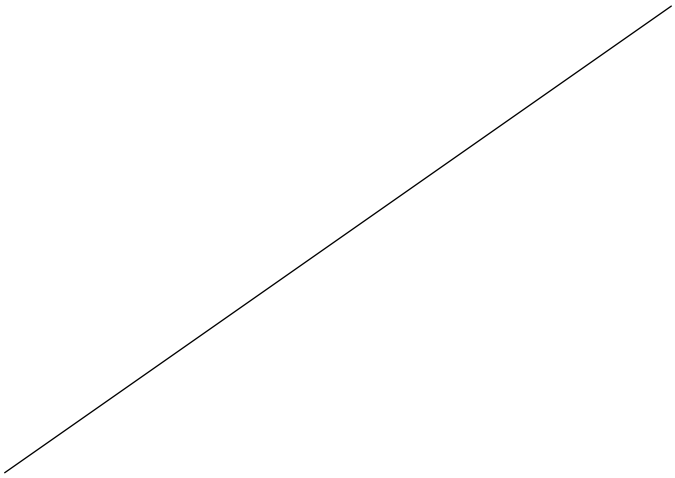
MetaPost: A Very Brief Tutorial

osurs@bluewin.ch Urs Oswald http://www.ursoswald.ch

October 3, 2002

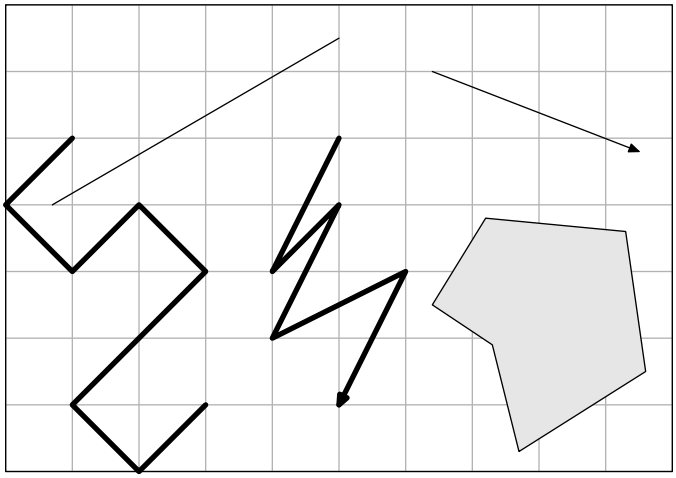
Contents

1 A2B



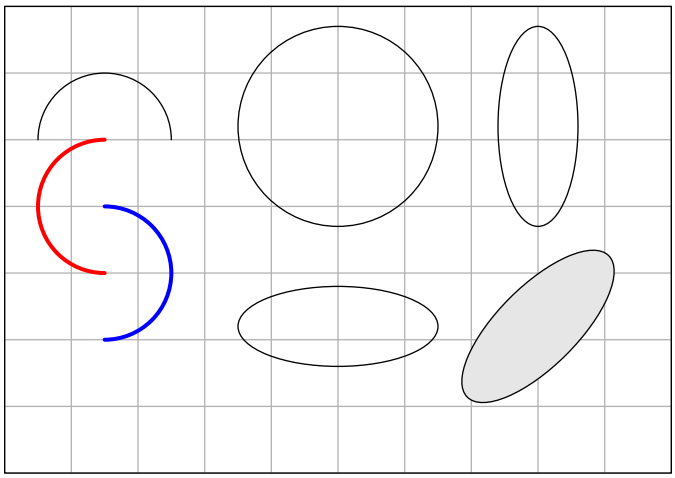
```
beginfig(1)
  draw (100, 100)--(350, 275);
endfig;

end
```

2 LineSegments



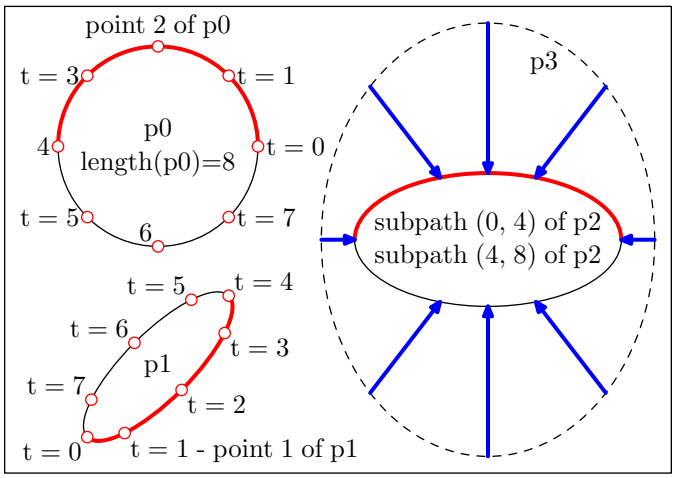
```
u:=25;                    % 25 = 25bp = 25 PostScript points = 25/72 in
wi:=10;                   % width  in units u
he:=7;                    % height in units u
hoehe:=he*u;              % height
breite:=wi*u;             % width

beginfig(1)
  % --- Grid ---
  for i=0 upto he:
    draw (0, i*u)--(breite, i*u) withcolor .7white;
  endfor
  for j=0 upto wi:
    draw (j*u, 0)--(j*u, hoehe) withcolor .7white;
  endfor
  % --- End Grid ---

  draw (0, 0)--(breite, 0)--(breite, hoehe)--(0, hoehe)--cycle;

  % --- Line Segment ---
  draw (.7u, 4u)--(5u, 6.5u);

  % --- Arrow ---
  drawarrow (6.4u, 6u)--(9.5u, 4.8u);

  pickup pencircle scaled 2;            % default:
                                        % 0.5 (= 0.5bp = 0.5 PostScript Points)
  % --- Polyline ---
  draw (u, 5u)--(0, 4u)--(u, 3u)--(2u, 4u)--(3u, 3u)--(u,u)--(2u, 0)--(3u, u);

  % --- Polyarrow ---
  drawarrow (5u, 5u)--(4u, 3u)--(5u, 4u)--(4u, 2u)--(6u, 3u)--(5u, u);

  % --- Polygon ---
  fill (7.7u, .3u)--(9.6u, 1.5u)--(9.3u, 3.6u)--(7.2u, 3.8u)
                 --(6.4u, 2.5u)--(7.3u, 1.9u)--cycle withcolor .9white;
  pickup pencircle scaled .5;
  draw (7.7u, .3u)--(9.6u, 1.5u)--(9.3u, 3.6u)--(7.2u, 3.8u)
       --(6.4u, 2.5u)--(7.3u, 1.9u)--cycle;
endfig;

end
```

3 Circles



```
%
%  /home/osurs/latex/metapost/TutorialAugsburg/tutorial/Circles/Circles.mp
%  18.09.02
%

u:=25;                    % 25 = 25bp = 25 PostScript points = 30/72 in
wi:=10;                   % width  in units u
he:=7;                    % height in units u
hoehe:=he*u;              % height
breite:=wi*u;             % width

beginfig(1)
  % --- Grid ---
  for i=0 upto he:
    draw (0, i*u)--(breite, i*u) withcolor .7white;
  endfor
  for j=0 upto wi:
    draw (j*u, 0)--(j*u, hoehe) withcolor .7white;
  endfor
  % --- End Grid ---

  draw (0, 0)--(breite, 0)--(breite, hoehe)--(0, hoehe)--cycle;

  %%%              %%%
  %%%  fullcircle  %%%
  %%%  halfcircle  %%%
  %%%              %%%
  draw fullcircle scaled 3u shifted (5u, 5.2u);
  draw fullcircle xscaled 3u yscaled 1.2u shifted (5u, 2.2u);
  draw fullcircle xscaled 1.2u yscaled 3u shifted (8u, 5.2u);
  fill fullcircle xscaled 3u yscaled 1.2u rotated 45 shifted (8u, 2.2u) withcolor .9white;
  draw fullcircle xscaled 3u yscaled 1.2u rotated 45 shifted (8u, 2.2u);

  draw halfcircle scaled 2u shifted (1.5u, 5u);
  pickup pencircle scaled 1.5;
  draw halfcircle scaled 2u rotated 90 shifted (1.5u, 4u) withcolor red;
  draw halfcircle scaled 2u rotated -90 shifted (1.5u, 3u) withcolor blue;
endfig;

end
```

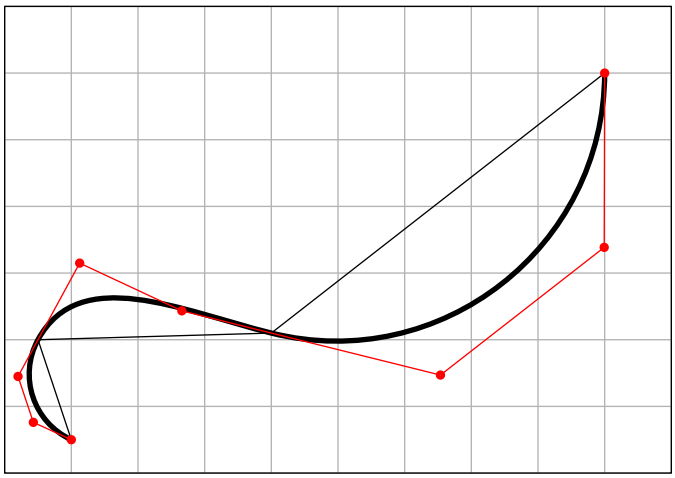
4 CirclePametrization




```
u:=25;                      % 25 = 25bp = 25 PostScript points = 25/72 in
wi:=10;                     % width  in units u
he:=7;                      % height in units u
hoehe:=he*u;                % height
breite:=wi*u;               % width

path p[];
  p0:=fullcircle scaled  3u shifted (2.3u, hoehe-2.1u);
  p1:=fullcircle xscaled 3u yscaled u rotated 225 shifted (2.3u, 1.6u);
  p2:=fullcircle xscaled 4u yscaled 2u shifted (7.25u, hoehe/2);
  p3:=fullcircle xscaled 5u yscaled 6.5u shifted (7.25u, hoehe/2);

def draw_point(expr P, colInt, colPer) =
  fill fullcircle scaled 1.5mm shifted P withcolor colInt;
  draw fullcircle scaled 1.5mm shifted P withcolor colPer;
enddef;

beginfig(1)
  % --- Draw frame ---
  draw (0, 0)--(breite, 0)--(breite, hoehe)--(0, hoehe)--cycle;

  draw p0;                                              % Draw p0
  pickup pencircle scaled 1.5;
  draw subpath (0, 4) of p0 withcolor red;
  pickup pencircle scaled 0.5;
  for t=0 upto 7:
    z[t]=point t of p0;
    draw_point(z[t], white, red);
    if (t<2) or (t=7):
      label.rt("t = "&decimal t, z[t]);
    elseif t=2:
      label.top("point "&decimal t&" of p0", z[t]);
    elseif (t=3) or (t=5):
      label.lft("t = "&decimal t, z[t]);
    elseif t=4:
      label.lft(decimal t, z[t]);
    else:
      label.ulft(decimal t, z[t]);
    fi
  endfor
  label("p0", center p0 + (0, .25u));
  label("length(p0)="&decimal length(p0), center p0 - (0, .25u));
```

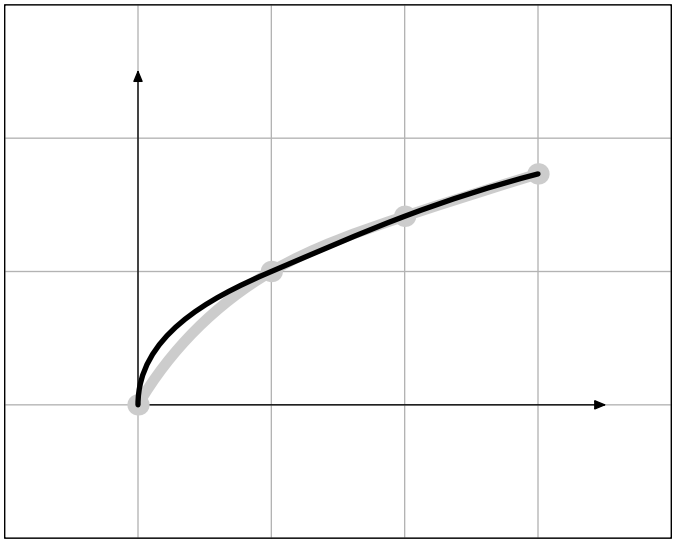
```
  draw p1;                                          % Draw p1
  pickup pencircle scaled 1.5;
  draw subpath (0, 4) of p1 withcolor red;
  pickup pencircle scaled 0.5;
  for t=0 upto 7:
    z[10+t]=point t of p1;
    draw_point(z[10+t], white, red);
    if t=0:
      label.llft("t = "&decimal t, z[10+t]);
    elseif t=1:
      label.lrt("t = "&decimal t&" - point "&decimal t&" of p1", z[10+t]);
    elseif t<4:
      label.lrt ("t = "&decimal t, z[10+t]);
    elseif t=4:
      label.urt ("t = "&decimal t, z[10+t]);
    else:
      label.ulft("t = "&decimal t, z[10+t]);
    fi
  endfor
  label("p1", center p1);

  draw p2;                                          % Draw p2
  pickup pencircle scaled 1.5;
  draw subpath (0, 4) of p2 withcolor red;
  pickup pencircle scaled 0.5;
  label("subpath (0, 4) of p2", center p2 + (0, .25u));
  label("subpath (4, 8) of p2", center p2 - (0, .25u));

  draw p3 dashed evenly;                            % Draw p3
  label.llft("p3", point 1.4 of p3);

  for t=0 upto 7:                                   % Draw blue arrows
    p99:=(point t of p3)--center p3;
    pickup pencircle scaled 1.5;
    if t<4:
      drawarrow p99 cutafter subpath (0, 4) of p2 withcolor blue;
    else:
      drawarrow p99 cutafter subpath (4, 8) of p2 withcolor blue;
    fi
  endfor
endfig;



end
```

5 BezierCurves



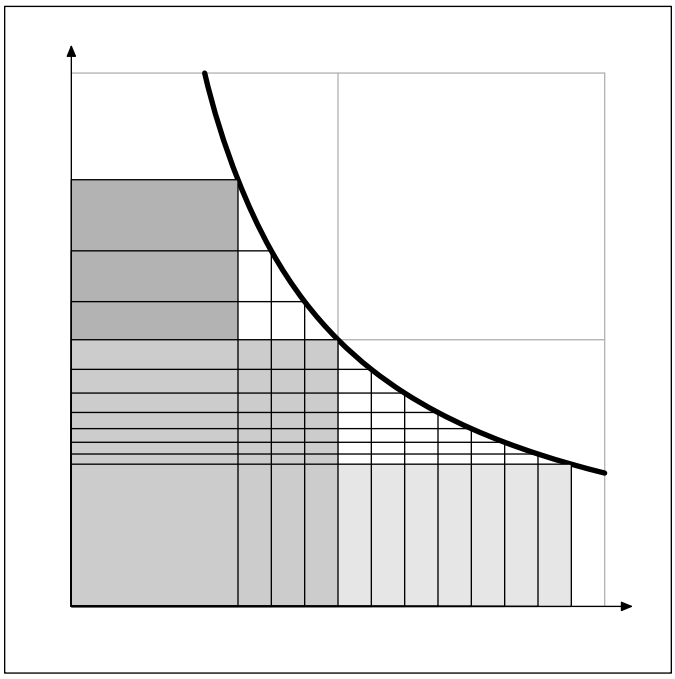
```
u:=25;                    % 25 = 25bp = 25 PostScript points = 25/72 in
wi:=10;                   % width  in units u
he:=7;                    % height in units u
hoehe:=he*u;              % height
breite:=wi*u;             % width

beginfig(1)
  % --- Draw Grid ---
  for i=0 upto he:
    draw (0, i*u)--(breite, i*u) withcolor .7white;
  endfor
  for j=0 upto wi:
    draw (j*u, 0)--(j*u, hoehe) withcolor .7white;
  endfor
  % --- End Grid ---

  % --- Draw Frame ---
  draw (0, 0)--(breite, 0)--(breite, hoehe)--(0, hoehe)--cycle;

  % --- Draw Line Segments ---
  % draw (u, .5u)--(.5u, 2u)--(4u, 2.7u)--(9u, 6u);
  draw (u, .5u)--(.5u, 2u)--(4u, 2.1u)--(9u, 6u);

  % --- Draw 3 Consecutive Cubic Bezier Curves ---
  pickup pencircle scaled 2;
  draw (u, .5u)..(.5u, 2u)..(4u, 2.1u)..(9u, 6u);

  % --- Draw Controls ---
  path p, q;
  p:=(u, .5u)..(.5u, 2u)..(4u, 2.1u)..(9u, 6u);
  q:=precontrol 0 of p--postcontrol 0 of p--
     precontrol 1 of p--postcontrol 1 of p--
     precontrol 2 of p--postcontrol 2 of p--
     precontrol 3 of p--postcontrol 3 of p;
  for i=0 upto 7:
    draw fullcircle scaled 1.5 shifted point i of q withcolor red;
  endfor
  pickup pencircle scaled .5;
  draw q withcolor red;
endfig;

end
```

6 GraphSqrt



```
u:=50;                    % 50 = 50bp = 50 PostScript points = 50/72 in
wi:=5;                    % width  in units u
he:=4;                    % height in units u
hoehe:=he*u;              % height
breite:=wi*u;             % width

beginfig(1)
  % --- Grid ---
  for i=0 upto he:
    draw (0, i*u)--(breite, i*u) withcolor .7white;
  endfor
  for j=0 upto wi:
    draw (j*u, 0)--(j*u, hoehe) withcolor .7white;
  endfor
  % --- End Grid ---

  draw (0, 0)--(breite, 0)--(breite, hoehe)--(0, hoehe)--cycle;

  z1=(0,    sqrt 0);
  z2=(1,    sqrt 1);
  z3=(2,    sqrt 2);
  z4=(3,    sqrt 3);

  drawarrow ((0,0)--(3.5, 0)) scaled u shifted (u, u);      % x-Achse
  drawarrow ((0,0)--(0,2.5)) scaled u shifted (u, u);       % y-Achse

  pickup pencircle scaled 4;
  draw (z1..z2..z3..z4) scaled u shifted (u, u) withcolor 0.8white;

  pickup pencircle scaled 8;
  for i=1 upto 4:
    draw z[i] scaled u shifted (u,u) withcolor .8white;
  endfor

  pickup pencircle scaled 2;
  draw (z1{up}..z2..z3..z4) scaled u shifted (u, u);

endfig;

end
```

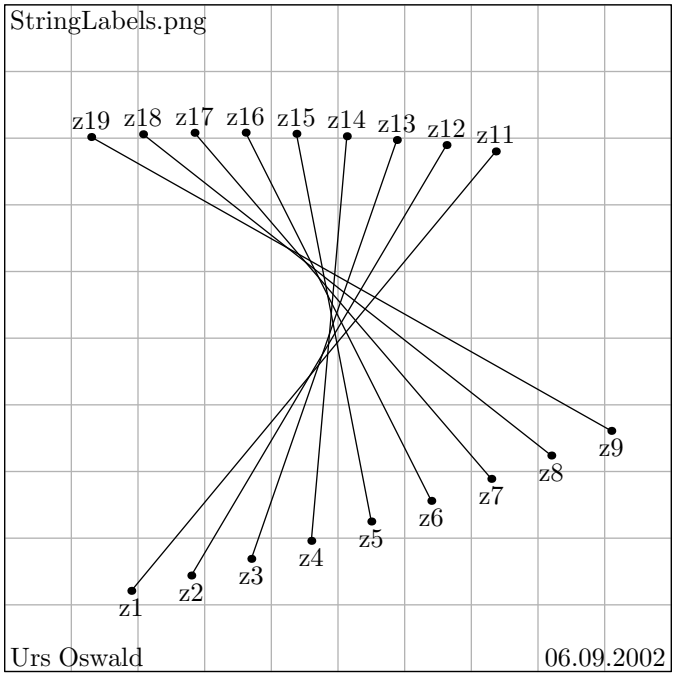
7 Paths



```
u:=100;                    % 100 = 100bp = 100 PostScript points = 100/72 in
hoehe:=2.5u;               % height
breite:=2.5*u;             % width
path h, r[];

beginfig(1)
  % --- Origin ---
  z0=(.25u, .25u);

  % --- Calculate 13 Points z4,...,z16 of Hyperbola y = 1/x ---
  for i=4 upto 16:
    z[i]=(i/8, 8/i);
    r[i]:=(0,0)--(x[i], 0)--z[i]--(0, y[i])--cycle;
  endfor

  % --- Form Path of Hyperbola From Points z5,...,z16 ---
  h:=(.5, 2) for i=5 upto 16: ..z[i] endfor ;

  % -- Draw Frame ---
  draw (0,0)--(breite,0)--(breite, hoehe)--(0, hoehe)--cycle;

  % --- Draw Grid ---
  for i=1,2:
    draw ((0,i)--(2,i)) scaled u shifted z0 withcolor .7white;
    draw ((i,0)--(i,2)) scaled u shifted z0 withcolor .7white;
  endfor
  % --- End Grid ---

  % --- Draw Axes ---
  drawarrow ((0, 0)--(2.1, 0)) scaled u shifted z0;
  drawarrow ((0, 0)--(0, 2.1)) scaled u shifted z0;

  % --- Draw Curve (Hyperbola) with pencircle scaled 2 ---
  pickup pencircle scaled 2;
  draw h scaled u shifted z0;
```

```
  % --- Fill Rectangles with colors .7white, .9white, and .8white ---
  fill r[5]  scaled u shifted z0 withcolor .7white;
  fill r[15] scaled u shifted z0 withcolor .9white;
  fill r[8]  scaled u shifted z0 withcolor .8white;

  % --- Draw Rectangles ---
  pickup pencircle scaled .5;      % 0.5: restore default pencircle
  for i=5 upto 15:
    draw r[i] scaled u shifted z0;
  endfor
endfig;

end
```

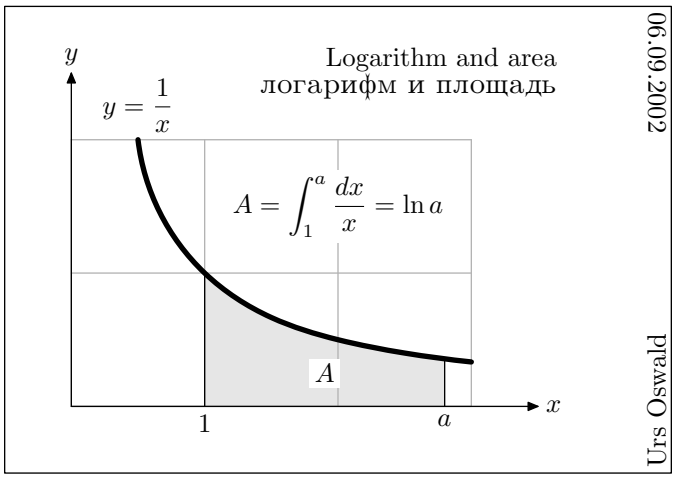
8 StringLabels




```
u:=25;                    % 25 = 25bp = 25 PostScript points = 30/72 in
wi:=10;                   % width  in units u
he:=10;                   % height in units u
hoehe:=he*u;              % height
breite:=wi*u;             % width
string s[];
s1:="06.09.2002";
s2:="StringLabels.png";

beginfig(1)
  % --- Grid ---
  for i=0 upto he:
    draw (0, i*u)--(breite, i*u) withcolor .7white;
  endfor
  for j=0 upto wi:
    draw (j*u, 0)--(j*u, hoehe) withcolor .7white;
  endfor
  % --- End Grid ---

  for i=1 upto 9:
    z[i]=(1+.9i, 1+.2i+.01i*i) scaled u;
    z[10+i]=z[i] rotatedaround((5u,5u), 160) scaled .8 yscaled 1.3;
  endfor

  % frame
  draw (0, 0)--(breite, 0)--(breite, hoehe)--(0, hoehe)--cycle;

  for i=1 upto 9:
    draw (z[i]--z[10+i]);                          % connections
  endfor

  for i=1 upto 9:
    dotlabel.bot("z"&decimal i, z[i]);             % bot: bottom
    dotlabel.top("z"&decimal (10+i), z[10+i]);     % &:   catenation of strings
  endfor
```

```
  label.urt("Urs Oswald", (0, 0));               % urt:  upper right
  label.ulft(s[1],        (breite, 0));          % ulft: upper left
  label.lrt(s[2],         (0, hoehe));           % lrt:  lower right
endfig;

end
```

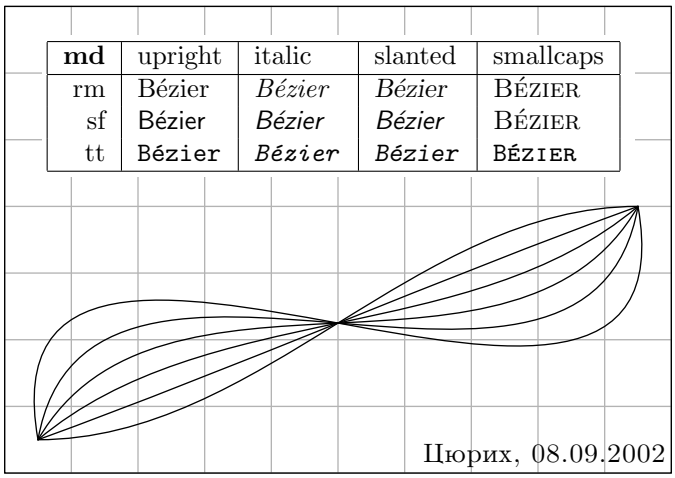
9 TeXLabels




```
verbatimtex
  \font\cyr=wncyr10
etex

u:=50;                    % 50 = 50bp = 50 PostScript points = 50/72 in
wi:=5;                    % width  in units u
he:=3.5;                  % height in units u
hoehe:=he*u;              % height
breite:=wi*u;             % width
path p, q[];
transform t;
picture lab[];

beginfig(1)
  z0=.5(u,u);
  t:=identity scaled u shifted z0;

  z1=(3.5, 0) transformed t;
  z2=(0, 2.5) transformed t;

  p:=(.5, 2) for i=2 upto 6: ..(.5i, 1/(.5i)) endfor ;
  p:=p transformed t;

  q0:=((1,   -0.5)--(1, 2.5))   transformed t;
  q1:=((2.8, -0.5)--(2.8, 2.5)) transformed t;

  q2:=buildcycle(z0--z1, q1, p, q0);

  lab0:=thelabel(btex $A$ etex, center q2 shifted (0, -u/4));
  lab1:=thelabel(btex $\displaystyle A=\int_1^a {dx\over x}=\ln a$ etex,
                   (2, 1.5) transformed t);

  % frame
  draw (0, 0)--(breite, 0)--(breite, hoehe)--(0, hoehe)--cycle;

  fill q2 withcolor .9white;

  % --- Grid ---
  for i=0 upto 2:
    draw ((0, i)--(3, i)) transformed t withcolor .7white;
  endfor
  for j=0 upto 3:
```

```
    draw ((j, 0)--(j, 2)) transformed t withcolor .7white;
  endfor
  % --- End Grid ---

  drawarrow z0--z1;                % x-Achse
  drawarrow z0--z2;                % y-Achse

  draw q0 cutbefore (z0--z1) cutafter p;
  draw q1 cutbefore (z0--z1) cutafter p;

  pickup pencircle scaled 2;
  draw p;

  % ============================== labels ===================================

  label.rt(btex $x$ etex, z1);
  label.top(btex $y$ etex, z2);

  label.bot(btex $1$ etex, (z0--z1) intersectionpoint q0);
  label.bot(btex $a$ etex, (z0--z1) intersectionpoint q1);

  label.top(btex $\displaystyle y={1\over x}$ etex, (.5, 2) transformed t);

  unfill bbox lab0; draw lab0;
  unfill bbox lab1; draw lab1;

  label.lft(btex Logarithm and area etex, (3.7, 2.6) transformed t);
  label.lft(btex
            \cyr  logarifm i plowad\char126
    etex scaled 1.1, (3.7, 2.4) transformed t);

  label.ulft(btex Urs Oswald etex rotated  90, (breite, 0));
  label.llft(btex 06.09.2002 etex rotated -90, (breite, hoehe));

  % ============================== labels ===================================

endfig;

end
```

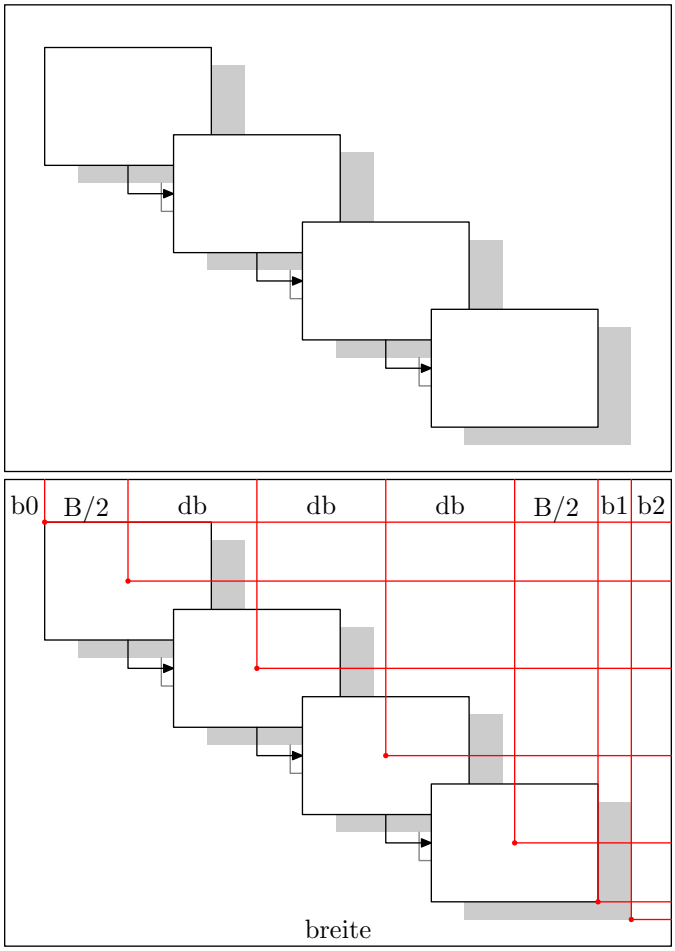
10 LaTeXLabels




```
verbatimtex
%&latex
\documentclass{article}
\newcommand{\uB}{\upshape{B\'ezier}}   % up: upright
\newcommand{\iB}{\itshape{B\'ezier}}   % it: italic
\newcommand{\lB}{\slshape{B\'ezier}}   % sl: slanted
\newcommand{\cB}{\scshape{B\'ezier}}   % sc: small caps
\newfont{\cyr}{wncyr10}
\begin{document}
etex

u:=25;                    % 25 = 25bp = 25 PostScript points = 30/72 in
wi:=10;                   % width  in units u
he:=7;                    % height in units u
hoehe:=he*u;              % height
breite:=wi*u;             % width
picture lab;

beginfig(1)
  % --- Grid ---
  for i=0 upto he:
    draw (0, i*u)--(breite, i*u) withcolor .7white;
  endfor
  for j=0 upto wi:
    draw (j*u, 0)--(j*u, hoehe) withcolor .7white;
  endfor
  % --- End Grid ---

  draw (0, 0)--(breite, 0)--(breite, hoehe)--(0, hoehe)--cycle;

  for i=0 upto 5:
    draw .5(u, u){dir 20i}..{dir 20i}(9.5u, 4u);
  endfor

  lab:=\thelabel(
    btex
      \begin{tabular}{|r|l|l|l|l|}
\hline
\textbf{md} & upright      & italic       & slanted      & smallcaps   \\
\hline
rm          & \textrm{\uB} & \textrm{\iB} & \textrm{\lB} &\textrm{\cB} \\
sf          & \textsf{\uB} & \textsf{\iB} &\textsf{\lB}  &\textsf{\cB} \\
```

```
tt          & \texttt{\uB} & \texttt{\iB} &\texttt{\lB}  &\texttt{\cB} \\
\hline
      \end{tabular}
    etex,
    (.5breite, hoehe-1.5u)
  );

  unfill bbox lab;
  draw lab;

  label.ulft(btex \cyr C\char24 rih, 08.09.2002 etex, (breite, 0));
endfig;

end
```

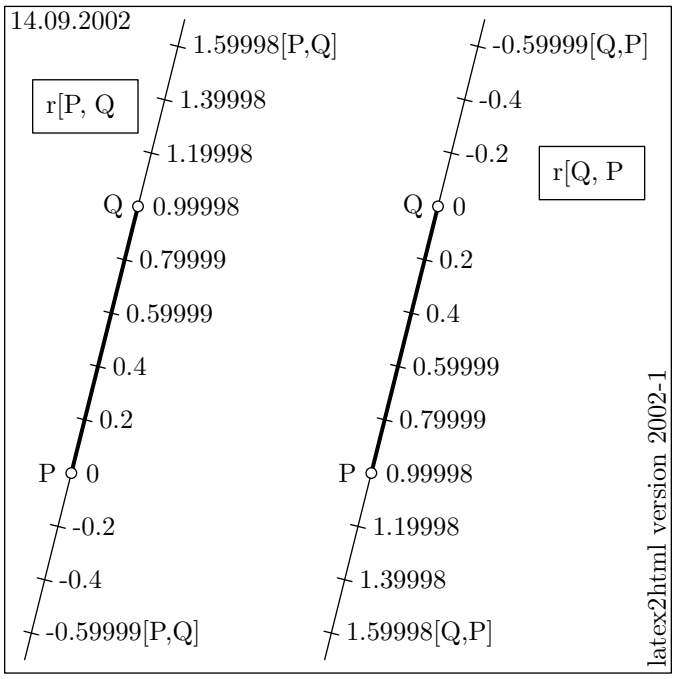
11 Equations




```
breite:=250;              % width
hoehe:=175;               % height

n:=4;
b0:=15;
b2:=15;
B:=.25breite;
h0:=10;
h2:=16;
H:=B/sqrt 2;

b1=.2B;
h1=.15H;

b0+B+(n-1)*db+b1+b2=breite;        % unknown db is calculated by MetaPost
h0+H+(n-1)*dh+h1+h2=hoehe;         % unknown dh is calculated by MetaPost

def draw_rect(expr P, arrow) =
  save p, q;
  path p, q;
  p:=P+.5(-B, -H)--P+.5(+B, -H)--P+.5(+B, +H)--P+.5(-B, +H)--cycle;
  q:=P+.5(0, -H)--P+(0, -dh)--P+(db-B/2, -dh);
  fill p shifted (b1, -h1) withcolor .8white;
  if arrow:
    drawarrow q shifted (b1, -h1) withcolor .5white;
  fi
  unfill p;
```

```
  draw p;
  if arrow:
    drawarrow q;
  fi
enddef;

beginfig(1)
  z0=(b0, hoehe-h2);
  for i=1 upto n:
    z[i]=(b0+B/2, hoehe-h2-H/2)+(i-1)*(db, -dh);
  endfor
  z[n+1]=z[n]+.5(B, -H);
  z[n+2]=z[n+1]+(b1, -h1);

  draw (0, 0)--(breite, 0)--(breite, hoehe)--(0, hoehe)--cycle;
  for i=1 upto n:
    draw_rect(z[i], i<n);
  endfor
endfig;

beginfig(2)
  z0=(b0, hoehe-h2);
  for i=1 upto n:
    z[i]=(b0+B/2, hoehe-h2-H/2)+(i-1)*(db, -dh);
  endfor
  z[n+1]=z[n]+.5(B, -H);
  z[n+2]=z[n+1]+(b1, -h1);

  draw (0, 0)--(breite, 0)--(breite, hoehe)--(0, hoehe)--cycle;
  for i=1 upto n:
    draw_rect(z[i], i<n);
  endfor
  for i=0 upto n+2:
    draw (x[i], hoehe)--z[i]--(breite, y[i]) withcolor red;
    fill fullcircle scaled 2 shifted z[i] withcolor red;
  endfor
  for i=0 upto n+2:
    if (i=0) or (i=n):
      if i=0:
        label.top("b0" , (.5[0, x[i]], hoehe-h2));
      fi
      label.top("B/2" , (.5[x[i], x[i+1]], hoehe-h2-3));
    elseif i<=n:
      label.top("db" , (.5[x[i],  x[i+1]], hoehe-h2));
    elseif i=n+1:
      label.top("b1" , (.5[x[i],  x[i+1]], hoehe-h2));
    else:
      label.top("b2" , (.5[x[i],  breite], hoehe-h2));
    fi
  endfor
  label.top("breite", (breite/2, 0));
endfig;

end
```

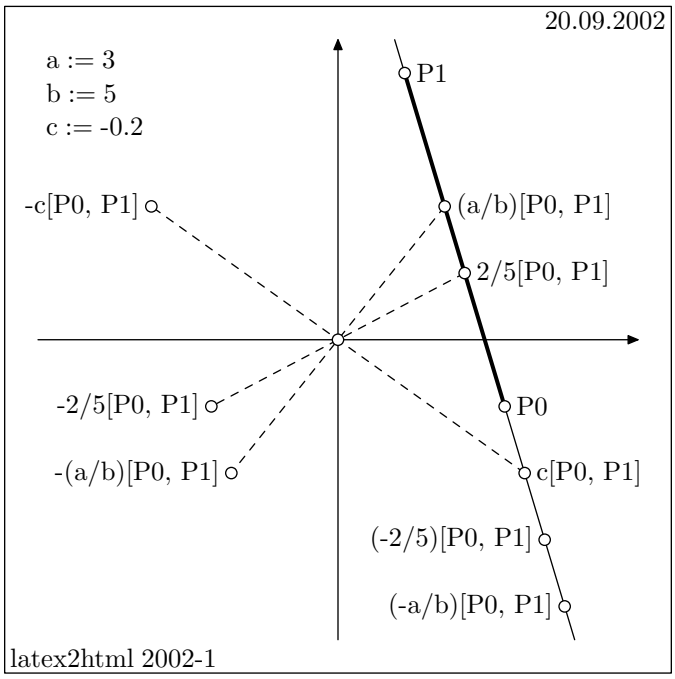
12 Mediation




```
u:=25;                    % 25 = 25bp = 25 PostScript points = 25/72 in
wi:=10;                   % width  in units u
he:=10;                   % height in units u
hoehe:=he*u;              % height
breite:=wi*u;             % width

marklength:=2mm;

transform t;
t:=identity scaled u;

def draw_point(expr P, colI, colP) =
  fill fullcircle scaled 1.4mm shifted P withcolor colI;
  draw fullcircle scaled 1.4mm shifted P withcolor colP;
enddef;

def mark_LineSegment(expr P, Q, ratio, c) =
  save q, E;
  path q; pair E;
  E:=unitvector(Q-P) scaled .5marklength;
  q:=E--E rotated 180;
  draw q rotated 90 shifted ratio[P, Q] withcolor c;
enddef;

def draw_MarkedMediationPoints(expr P, Q, reverse) =
  %
  % draw marks at mediation points
  %
  pair Z[];
  string s[];
  Z0=P transformed t;
  Z1=Q transformed t;
  if reverse:
    s0:="Q"; s1:="P";
  else:
    s0:="P"; s1:="Q";
```

```
  fi
  pickup pencircle scaled 1.5;
  draw Z0--Z1;                                    % Draw line segment
  pickup pencircle scaled 0.5;
  draw (-.7)[Z0, Z1]--(1.7)[Z0, Z1];              % Draw straight line
  for i=-3 upto 8:                                % prolonging line segment
    ratio:=.2i;                                   % ratio = -.6, -.4, ..., 1.4, 1.6
    if (i<>0) and (i<>5):
      mark_LineSegment(Z0, Z1, ratio, black);     % draw mark if i not equal to
    fi                                            % either 0 or 5 (ratios 0 and 1)
    if (i=-3) or (i=8):
      label.rt((decimal ratio)&"["&s0&","&s1&"]",
        ratio[Z0, Z1]+(.1u,0));
    else:
      label.rt(decimal ratio, ratio[Z0, Z1]+(.1u,0));
    fi
    if i=0:
      label.lft(s0, ratio[Z0, Z1]-(.1u,0));
    fi
    if i=5:
      label.lft(s1, ratio[Z0, Z1]-(.1u,0));
    fi
  endfor
  pickup pencircle scaled 0.5;
  draw_point(Z0, white, black);                   % Draw endpoints
  draw_point(Z1, white, black);
enddef;

beginfig(1)
  bboxmargin:=5;
  pair P, Q, L, versch;
  picture lab;
  P:=(1, 3);                                      % starting point of line segment
  Q:=(2, 7);                                      % end point of line segment
  L:=(0.5, 8.5);                                  % label
  versch:=(4.5, 0);

  % --- Draw Frame ---
  draw (0, 0)--(breite, 0)--(breite, hoehe)--(0, hoehe)--cycle;

  lab:=\thelabel.rt("r[P, Q]", L transformed t);  % draw left label
  unfill bbox lab; draw lab; draw bbox lab;

  draw_MarkedMediationPoints(P, Q, false);        % false: not reverse
                                                  % i.e. mediation order P, Q
  P:=P shifted versch;
  Q:=Q shifted versch;
  L:=L shifted versch+(3.1,-1);

  lab:=\thelabel.rt("r[Q, P]", L transformed t);  % draw right label
  unfill bbox lab; draw lab; draw bbox lab;

  draw_MarkedMediationPoints(Q, P, true);         % true: reverse
                                                  % i.e. mediation order Q, P

  label.lrt (btex 14.09.2002 etex, (0, hoehe));
  label.ulft(btex latex2html version 2002-1 etex rotated 90, (breite, 0));
endfig;

end
```

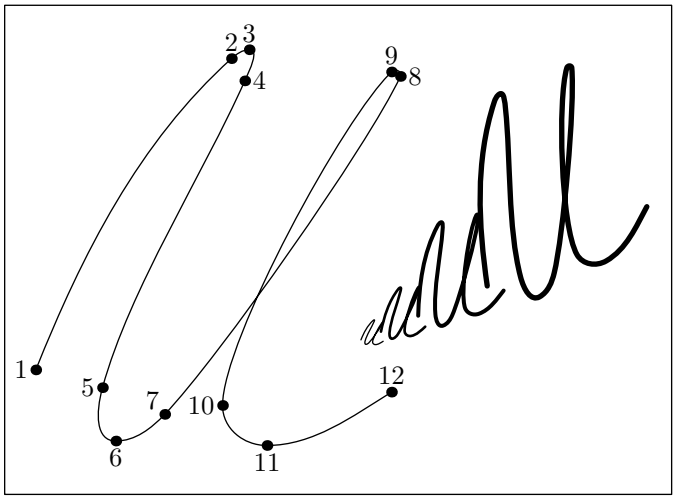
13 Precedence




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

u:=25;                    % 25 = 25bp = 25 PostScript points = 25/72 in
wi:=10;                   % width  in units u
he:=10;                   % height in units u
hoehe:=he*u;              % height
breite:=wi*u;             % width
pair LL, P[];
LL:=(-5, -5);             % LL: Lower Left
P0:=(2.5, -1)*u;          % P0: Starting point of line segment
P1:=(1, 4)*u;             % P1: End point of line segment

a:=3; b=5; c:=-1/5;

def draw_point(expr P, colInt, colPer) =
  fill fullcircle scaled 1.5mm shifted P withcolor colInt;
  draw fullcircle scaled 1.5mm shifted P withcolor colPer;
enddef;

beginfig(1)
  % --- Calculate endpoints of prolongation of line segment PQ ---
  z0=whatever[P0, P1]=whatever[(0, u*(ypart LL+.5)), (1, u*(ypart LL+.5))];
  z1=whatever[P0, P1]=whatever[(0, u*(he+ypart LL-.5)), (1, u*(he+ypart LL-.5))];

  % --- Draw Frame ---
  draw (LL--(LL+(wi, 0))--(LL+(wi, he))--(LL+(0, he))--cycle) scaled u;

  % --- Draw axes ---
  drawarrow ((xpart LL+.5, 0)--(xpart LL+wi-.5, 0)) scaled u;
  drawarrow ((0, ypart LL+.5)--(0, ypart LL+he-.5)) scaled u;

  %--- Draw PQ and prolongation ---
```

```
  pickup pencircle scaled 1.5; draw P0--P1;
  pickup pencircle scaled .5;  draw z0--z1;

  % --- Show values of a, b, c ---
  label.rt("a := "&decimal a, ((.5, he-0.8)+LL)*u);
  label.rt("b := "&decimal b, ((.5, he-1.3)+LL)*u);
  label.rt("c := "&decimal c, ((.5, he-1.8)+LL)*u);

  % --- Draw connections of points reflected at the origin ---
  draw 2/5[P0, P1]--(-2/5[P0, P1]) dashed evenly;
  draw (a/b)[P0, P1]--(-(a/b)[P0, P1]) dashed evenly;
  draw c[P0, P1]--(-c[P0, P1]) dashed evenly;

  % --- Draw end points of given line segment ---
  draw_point(P0, white, black); label.rt("P0", P0+(.5mm,0));
  draw_point(P1, white, black); label.rt("P1", P1+(.5mm,0));

  % --- Draw mediation points ---
  draw_point(2/5[P0, P1],    white, black);
  label.rt (btex    2/5[P0, P1] etex,    2/5[P0, P1]+( .5mm, 0));
  draw_point(-2/5[P0, P1],   white, black);
  label.lft(btex   -2/5[P0, P1] etex,   -2/5[P0, P1]-( .5mm, 0));
  draw_point((-2/5)[P0, P1], white, black);
  label.lft(btex (-2/5)[P0, P1] etex, (-2/5)[P0, P1]-( .5mm, 0));
  draw_point((a/b)[P0, P1],  white, black);
  label.rt (btex  (a/b)[P0, P1] etex,  (a/b)[P0, P1]+( .5mm, 0));
  draw_point(-(a/b)[P0, P1], white, black);
  label.lft(btex -(a/b)[P0, P1] etex, -(a/b)[P0, P1]+(-.5mm, 0));
  draw_point((-a/b)[P0, P1], white, black);
  label.lft(btex (-a/b)[P0, P1] etex, (-a/b)[P0, P1]+(-.5mm, 0));
  draw_point(c[P0, P1],      white, black);
  label.rt (btex      c[P0, P1] etex,      c[P0, P1]+( .5mm, 0));
  draw_point(-c[P0, P1],     white, black);
  label.lft(btex     -c[P0, P1] etex,     -c[P0, P1]-( .5mm, 0));

  draw_point((0,0),          white, black);

  % --- Draw labels ---
  label.urt("latex2html 2002-1", LL*u);
  label.llft("20.09.2002", (LL+(wi, he))*u);
endfig;

end
```

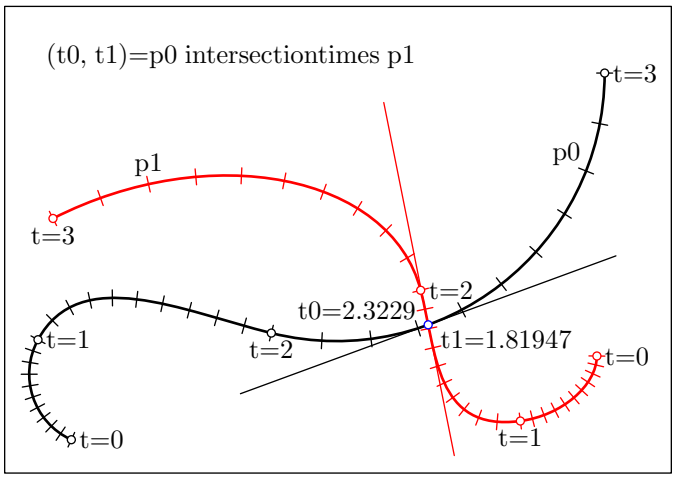
14 Directions




```
u:=40mm/68;
breite:=150u;
hoehe:=110u;

path p;

beginfig(1)
  z0=(.5breite, .3hoehe);

  z1=(7,28)*u;
  z2=(51,98)*u;
  z3=(55,100)*u;
  z4=(54,93)*u;
  z5=(22,24)*u;
  z6=(25,12)*u;
  z7=(36,18)*u;
  z8=(89,94)*u;
  z9=(87,95)*u;
  z10=(49,20)*u;
  z11=(59,11)*u;
  z12=(87,23)*u;

  p:=z1{2,5}..
     z2{10,9}..
     z3{right}..tension 1.3..
     z4{-4,-9}..tension 1.1..
     z5{-17,-62}..
     z6{right}..
     z7..tension 1.8..
     z8{27,58}..
     z9{-55,-52}..tension 2..
     z10{down}..
     z11{right}..
     z12{50,31};

  draw (0,0)--(breite,0)--(breite,hoehe)--(0,hoehe)--cycle;

  draw p;                                    % original curve
  for i=1 upto 12:                           % labels 1 to 12
    if (i=2) or (i=3) or (i=9) or (i=12):
      label.top(decimal(i), z[i]);
    elseif (i=4) or (i=8):
```

```
      label.rt(decimal(i), z[i]);
    elseif (i=6) or (i=11):
      label.bot(decimal(i), z[i]);
    elseif (i=7):
      label.ulft(decimal(i), z[i]);
    else:
      label.lft(decimal(i), z[i]);
    fi
  endfor

  pickup pencircle scaled 4;                % dot labels
  for i=1 upto 12:
    draw z[i];
  endfor

                                            % transform original curve
  pickup pencircle scaled .5;
  draw p rotated   0 shifted (.5breite, 0) scaled 1/16 shifted z0;
  pickup pencircle scaled 1;
  draw p rotated  10 shifted (.5breite, 0) scaled 1/8 shifted z0;
  pickup pencircle scaled 1.5;
  draw p rotated  20 shifted (.5breite, 0) scaled 1/4 shifted z0;
  pickup pencircle scaled 2;
  draw p rotated  30 shifted (.5breite, 0) scaled 1/2 shifted z0;

endfig;

end
```

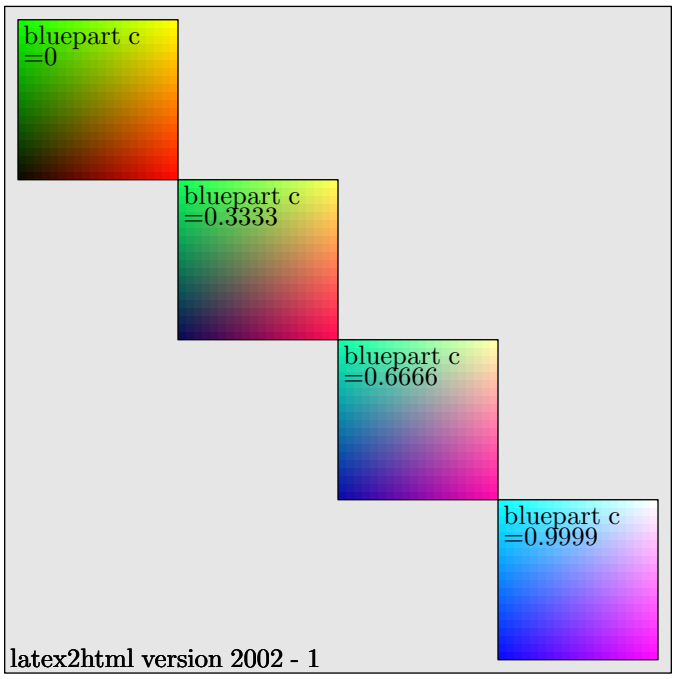
15 Times




```
u:=25;                    % 25 = 25bp = 25 PostScript points = 25/72 in
wi:=10;                   % width  in units u
he:=7;                    % height in units u
hoehe:=he*u;              % height
breite:=wi*u;             % width
path p[];
pair P;

def draw_point(expr P, colI, colP) =
  fill fullcircle scaled 3 shifted P withcolor colI;
  draw fullcircle scaled 3 shifted P withcolor colP;
enddef;

beginfig(1)
  z0=(1, .5)*u;
  z1=(.5, 2)*u;
  z2=(4, 2.1)*u;
  z3=(9, 6)*u;

  p0:=z0..z1..z2..z3;
  p1:=p0 slanted .2
           xscaled .7
     rotatedabout(.5(breite, hoehe), -55)
       shifted (.7u, 0)
         rotatedaround((.5(breite, hoehe)), 180)
   shifted (-.3u, 0);

  % --- Draw Frame ---
  draw (0, 0)--(breite, 0)--(breite, hoehe)--(0, hoehe)--cycle;
  %
  % --- Draw p0 withcolor black ---
  %
  pickup pencircle scaled 1;
  draw p0;
  pickup pencircle scaled .5;
  for i=0 upto 30:
    t2:=.1i;
    draw ((-3, 0)--(3, 0))rotated (90+angle direction t2 of p0)
      shifted point t2 of p0;
  endfor
  label.ulft("p0", point 2.8 of p0);
  %
```

```
  % --- Draw p1 withcolor red ---
  %
  pickup pencircle scaled 1;
  draw p1 withcolor red;
  pickup pencircle scaled .5;
  for i=0 upto 30:
    t2:=.1i;
    draw ((-3, 0)--(3, 0))rotated (90+angle direction t2 of p1)
      shifted point t2 of p1 withcolor red;
  endfor
  label.top("p1", point 2.8 of p1);
  %
  % --- Calculate and draw point of intersection and tangents
  %
  (t0, t1)=p0 intersectiontimes p1;
  P:=p0 intersectionpoint p1;
  draw ((-75, 0)--(75, 0)) rotated angle direction t0 of p0 shifted P;
  draw ((-50, 0)--(85, 0)) rotated angle direction t1 of p1 shifted P withcolor red;
  label.ulft("t0="&decimal t0, P-(.1u, 0));
  label.lrt("t1="&decimal t1, P+(.1u, 0));
  draw_point(P, white, blue);
  %
  % --- Draw points with integer parameter values ---
  %
  for i=0 upto 3:
    draw_point(point i of p0, white, black);
    if i<>2:
      label.rt("t="&decimal i, point i of p0);
    else:
      label.bot("t="&decimal i, point i of p0);
    fi
  endfor
  for i=0 upto 3:
    draw_point(point i of p1, white, red);
    if (i=1) or (i=3):
      label.bot("t="&decimal i, point i of p1);
    else:
      label.rt("t="&decimal i, point i of p1);
    fi
  endfor

  label.rt("(t0, t1)=p0 intersectiontimes p1" ,(.5u, hoehe-.75u));

endfig;

end
```

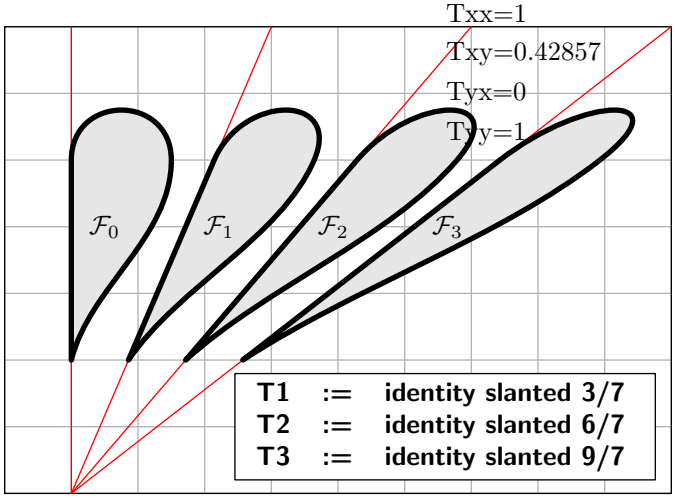
16 Colors




```
u:=25;                    % 25 = 25bp = 25 PostScript points = 25/72 in
wi:=10;                   % width  in units u
he:=10;                   % height in units u
hoehe:=he*u;              % height
breite:=wi*u;             % width
pair P[], xy[];
path p[]; transform t; color c[];
n:=20; rd:=0.2; b:=(wi-2rd)/4;

P0:=(0, 0);
t:=identity scaled u shifted P0;

beginfig(1)
  fill (0, 0)--(breite, 0)--(breite, hoehe)--(0, hoehe)--cycle withcolor .9white;
  draw (0, 0)--(breite, 0)--(breite, hoehe)--(0, hoehe)--cycle;

  for k=0 upto 3:
    blau:=.3333k;
    P1:=(rd, he-rd-b)+k*b*(1, -1);
    for i=0 upto n-1:
      for j=0 upto n-1:
        xy0:=P1+(i*b/n, j*b/n);
        p0:=xy0--(xy0+(b/n, 0))--(xy0+(b/n, b/n))--(xy0+(0, b/n))--cycle;
        c0:=((i+1)/n, (j+1)/n, blau);
        fill p0 transformed t withcolor c0;
      endfor
    endfor
    draw (P1--(P1+(b, 0))--(P1+(b, b))--(P1+(0,b))--cycle) transformed t;
    P2:=(P1+(0, b))transformed t;
    label.lrt("bluepart c", P2);
    label.lrt("="&decimal blau, P2+(0, -3mm));
    label.urt("latex2html version 2002 - 1", (0,0));
  endfor
endfig;

end
```

17 Slanted

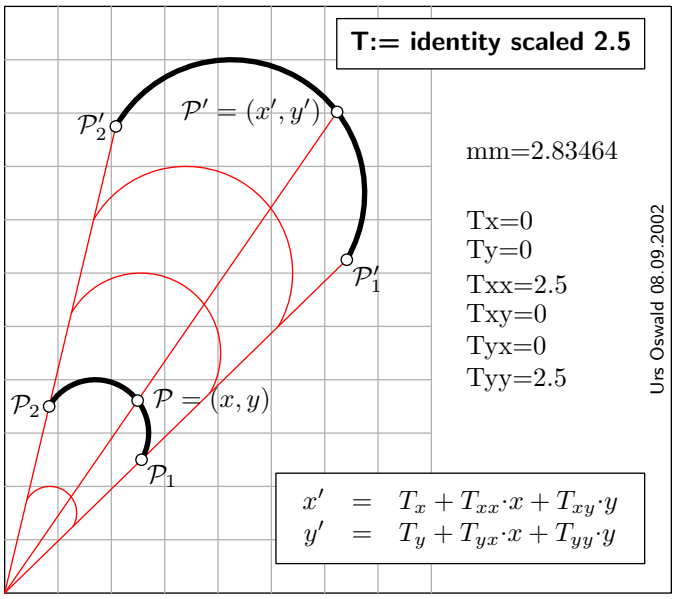
Tx=0
Ty=0
Txx=1
Txy=0.42857
Tyx=0
Tyy=1




```
verbatimtex
%&latex
\documentclass{article}
\newcommand{\fett}{\sffamily\bfseries}
\begin{document}
etex

u:=25;                    % 25 = 25bp = 25 PostScript points = 30/72 in
wi:=10;                   % width  in units u
he:=7;                    % height in units u
hoehe:=he*u;              % height
breite:=wi*u;             % width
path p[], q[];
picture pic;
transform t, T[];
t:=identity scaled u;

T1:=identity slanted 3/7;
T2:=identity slanted 6/7;
T3:=identity slanted 9/7;

beginfig(1)
  z0=(1,   2) transformed t;
  z1=(2.5, 5) transformed t;
  z2=(1,   5) transformed t;
  z3=(1.5, 4) transformed t;       % label

  p0:=z0{1,4}..z1{up}..{down}z2--cycle;
  q0:=(x0, 0)--(x0, hoehe);

  for i=1 upto 3:
    p[i]:=p0 transformed T[i];
    q[i]:=q0 transformed T[i];
  endfor

  % --- Grid ---
  for i=0 upto he:
    draw (0, i*u)--(breite, i*u) withcolor .7white;
  endfor
```

```
  for j=0 upto wi:
    draw (j*u, 0)--(j*u, hoehe) withcolor .7white;
  endfor
  % --- End Grid ---

  draw (0, 0)--(breite, 0)--(breite, hoehe)--(0, hoehe)--cycle;

  for i=0 upto 3:
    fill p[i] withcolor .9white;
    pickup pencircle scaled .5;
    draw q[i] withcolor red;
    pickup pencircle scaled 2;
    draw p[i];
  endfor

  label(btex $\mathcal{F}_0$ etex, z3);
  label(btex $\mathcal{F}_1$ etex, z3 transformed T1);
  label(btex $\mathcal{F}_2$ etex, z3 transformed T2);
  label(btex $\mathcal{F}_3$ etex, z3 transformed T3);

  pic:=\thelabel.lft(btex
                       \begin{tabular}{ccl}
 \fett T1 & \fett := & \fett identity slanted 3/7 \\
 \fett T2 & \fett := & \fett identity slanted 6/7 \\
 \fett T3 & \fett := & \fett identity slanted 9/7
       \end{tabular}
                     etex, (breite-.2u, 1u));

  unfill bbox pic;
  draw pic;
  pickup pencircle scaled .5;
  draw bbox pic;

  label.rt ("Tx="&decimal(xpart  T1), (6.5, 8.4) transformed t);
  label.rt ("Ty="&decimal(ypart  T1), (6.5, 7.8) transformed t);
  label.rt("Txx="&decimal(xxpart T1), (6.5, 7.2) transformed t);
  label.rt("Txy="&decimal(xypart T1), (6.5, 6.6) transformed t);
  label.rt("Tyx="&decimal(yxpart T1), (6.5, 6.0) transformed t);
  label.rt("Tyy="&decimal(yypart T1), (6.5, 5.4) transformed t);


endfig;

end
```

18 Scaled

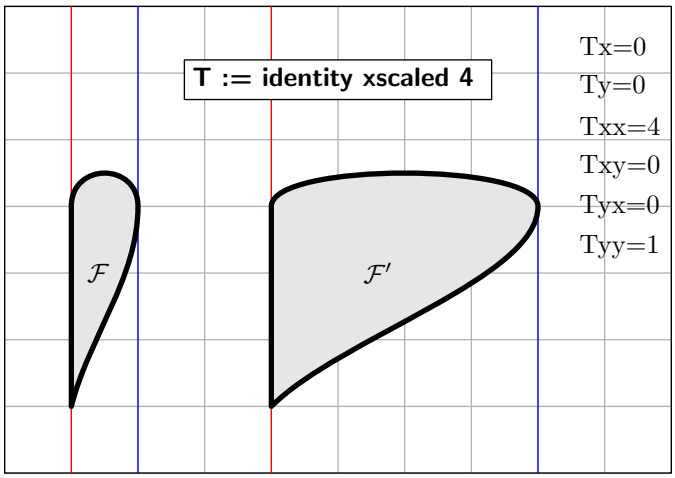



```
verbatimtex
%&latex
\documentclass{article}
\newcommand{\klein}{\sffamily\scriptsize}
\begin{document}
etex

def draw_point(expr P) =
  save r;
  r:=1.5mm;
  unfill fullcircle scaled r shifted P;
  draw   fullcircle scaled r shifted P;
enddef;

u:=20;
breite:=12.5u;
hoehe:=11u;
path p[];
picture pic[];
transform t, T;
T:=identity scaled 2.5;
bboxmargin:=5;                          % internal variable bboxmargin: default is 2 (bp)

beginfig(1)
  z0=(0,0);                             % z0: origin
  t:=identity scaled u;
  z1=(8.5, 7) transformed t;        % z1: top left, list of transform components

  p1:=halfcircle scaled 2 rotated -30 shifted (1.7 ,3);
  p1:=p1 transformed t;

  p2:=p1 transformed T;

  z11=point 0 of p1;
  z12=point infinity of p1;
  z10=point 1.5 of p1;

  z21=z11 transformed T;
  z22=z12 transformed T;
```

```
z20=z10 transformed T;

pic1:=\thelabel.lft(btex \sffamily\bfseries T:= identity scaled 2.5 etex,
                                    (breite-.6u, 10.3u));
%
% The \cdot command doesn't work with LaTeX2html version 2002 - 1
%
pic2:=\thelabel.lft(
  btex $
    \begin{array}{rcl}
      x' & = & T_x + T_{xx}\raisebox{.5ex}{.}x + T_{xy}\raisebox{.5ex}{.}y \\
      y' & = & T_y + T_{yx}\raisebox{.5ex}{.}x + T_{yy}\raisebox{.5ex}{.}y
    \end{array}
  $ etex,
  (breite-.6u, 1.4u)
);

% frame
draw (0, 0)--(breite, 0)--(breite, hoehe)--(0, hoehe)--cycle;

% grid
for i=1 upto 10:
  draw ((0, i)--(8, i)) transformed t withcolor .7white;
endfor
for j=1 upto 8:
  draw ((j, 0)--(j, 11)) transformed t withcolor .7white;
endfor

% scaling
draw z0--z21 withcolor red;
draw z0--z22 withcolor red;
draw z0--z20 withcolor red;
for i=1 upto 5:
  draw p1 scaled .5i withcolor red;
endfor

pickup pencircle scaled 2;
draw p1;
draw p2;

label.lft(btex \klein Urs Oswald 08.09.2002 etex rotated 90, (breite, .5hoehe));
label.lrt(btex $\mathcal{P}_1$ etex, z11);
label.lft(btex $\mathcal{P}_2$ etex, z12);
    %
    % mm is a purely numerical value:
    % It is the ratio of 1mm to 1bp.
    %
label.rt(btex $\mathcal{P}=(x,y)$   etex, z10+mm*(1, 0));

label.lrt(btex $\mathcal{P}_1'$ etex, z21);
label.lft(btex $\mathcal{P}_2'$ etex, z22);
label.lft(btex $\mathcal{P}'=(x',y')$ etex, z20-mm*(1,0));

pickup pencircle scaled .5;
for i=11, 12, 10, 21, 22, 20:
  draw_point(z[i]);
endfor

%
% Strings can be catenated by the operator `&'.
```

```
  %
  label.rt("mm="&decimal mm,         z1+(0, 1.3u));

  %
  % Show the 6 components xpart, ypart, xxpart, xypart, yxpart, yypart
  % of transformation T
  %
  label.rt ("Tx="&decimal(xpart  T), z1);  y1:=y1-.6u;
  label.rt ("Ty="&decimal(ypart  T), z1);  y1:=y1-.6u;
  label.rt("Txx="&decimal(xxpart T), z1);  y1:=y1-.6u;
  label.rt("Txy="&decimal(xypart T), z1);  y1:=y1-.6u;
  label.rt("Tyx="&decimal(yxpart T), z1);  y1:=y1-.6u;
  label.rt("Tyy="&decimal(yypart T), z1);  y1:=y1-.6u;

  unfill bbox pic1; draw pic1; draw bbox pic1;
  unfill bbox pic2; draw pic2; draw bbox pic2;

endfig;

end
```

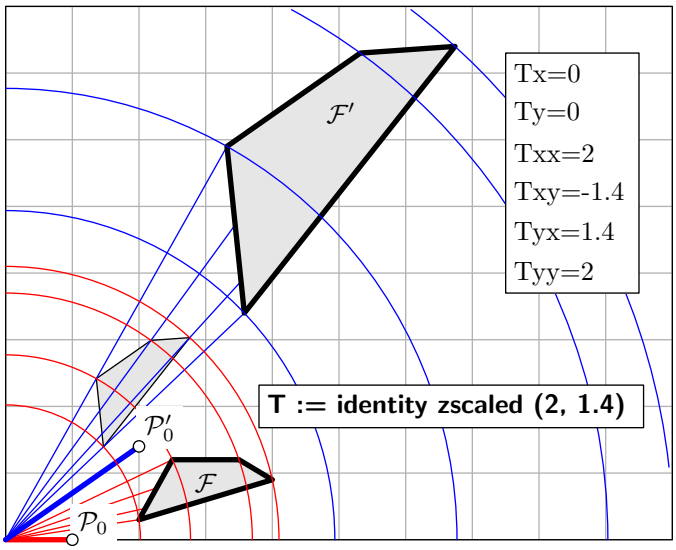
19 xScaled




```
verbatimtex
%&latex
\documentclass{article}
\newcommand{\fett}{\sffamily\bfseries}
\begin{document}
etex

u:=25;                    % 25 = 25bp = 25 PostScript points = 30/72 in
wi:=10;                   % width  in units u
he:=7;                    % height in units u
hoehe:=he*u;              % height
breite:=wi*u;             % width
path p[], q[], r[];
picture pic;
transform t, T;
t:=identity scaled u;
T:=identity xscaled 4;
bboxmargin:=3;

beginfig(1)
  z0=(1.0, 1) transformed t;
  z1=(2,   4) transformed t;
  z2=(1.0, 4) transformed t;
  z3=(1.4, 3) transformed t;       % label

  p0:=z0{1,4}..z1{up}..{down}z2--cycle;
  q0:=(x0, 0)--(x0, hoehe);
  r0:=(x1, 0)--(x1, hoehe);

    p1:=p0 transformed T;
    q1:=q0 transformed T;
    r1:=r0 transformed T;

  % --- Grid ---
  for i=0 upto he:
    draw (0, i*u)--(breite, i*u) withcolor .7white;
  endfor
  for j=0 upto wi:
    draw (j*u, 0)--(j*u, hoehe) withcolor .7white;
  endfor
  % --- End Grid ---
```

```
  draw (0, 0)--(breite, 0)--(breite, hoehe)--(0, hoehe)--cycle;

  for i=0 upto 1:
    fill p[i] withcolor .9white;
    pickup pencircle scaled .5;
    draw q[i] withcolor red;
    draw r[i] withcolor blue;
    pickup pencircle scaled 2;
    draw p[i];
  endfor

  label(btex $\mathcal{F}$ etex, z3);
  label(btex $\mathcal{F}'$ etex, z3 transformed T);

  if false:
    label(btex $\mathcal{F}_2$ etex, z3 transformed T2);
    label(btex $\mathcal{F}_3$ etex, z3 transformed T3);
  fi

  pic:=\thelabel(btex \fett T := identity xscaled 4
                    etex, (.5breite, hoehe-1.1u));

  unfill bbox pic;
  draw pic;
  pickup pencircle scaled .5;
  draw bbox pic;

  label.rt ("Tx="&decimal(xpart  T), (8.5, 6.4) transformed t);
  label.rt ("Ty="&decimal(ypart  T), (8.5, 5.8) transformed t);
  label.rt("Txx="&decimal(xxpart T), (8.5, 5.2) transformed t);
  label.rt("Txy="&decimal(xypart T), (8.5, 4.6) transformed t);
  label.rt("Tyx="&decimal(yxpart T), (8.5, 4.0) transformed t);
  label.rt("Tyy="&decimal(yypart T), (8.5, 3.4) transformed t);

endfig;

end
```

20 zScaled

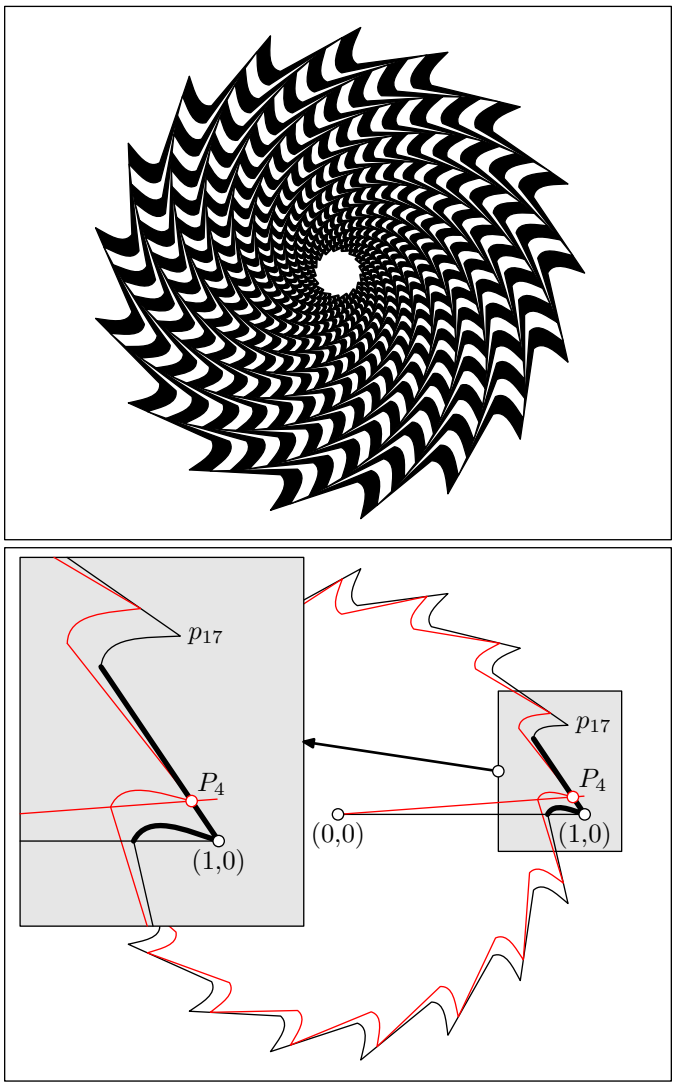



```
verbatimtex
%&latex
\documentclass{article}
\newcommand{\fett}{\sffamily\bfseries}
\begin{document}
etex

def draw_point(expr P) =
  save r;
  r:=1.5mm;
  unfill fullcircle scaled r shifted P;
  draw   fullcircle scaled r shifted P;
enddef;

u:=25;                    % 25 = 25bp = 25 PostScript points = 30/72 in
wi:=10;                   % width  in units u
he:=8;                    % height in units u
hoehe:=he*u;              % height
breite:=wi*u;             % width
path p[], q[], r[], c[];
picture pic[];
transform t, T;
t:=identity scaled u;

pair Pr[];
Pr0:=(2, 1.4);

T:=identity zscaled Pr0;
bboxmargin:=3;

beginfig(1)
  z0=(1,0) transformed t;

  % Polygon
  z1=(2.0, 0.3) transformed t;
  z2=(4,   0.9) transformed t;
  z3=(3.5, 1.2)  transformed t;
  z4=(2.5, 1.2)  transformed t;

  % Arcs
```

```
for i=1 upto 4:
  c[i]:=halfcircle scaled (2abs z[i]) cutafter ((0,0)--(0, hoehe));
endfor

% Polygons
p0:=z1--z2--z3--z4--cycle;
p1:=p0 transformed T;

for i=1 upto 4:
  q[i]:=(0,0)--z[i];
  r[i]:=q[i] transformed T;
endfor

% --- Grid ---
for i=0 upto he:
  draw (0, i*u)--(breite, i*u) withcolor .7white;
endfor
for j=0 upto wi:
  draw (j*u, 0)--(j*u, hoehe) withcolor .7white;
endfor
% --- End Grid ---

draw (0, 0)--(breite, 0)--(breite, hoehe)--(0, hoehe)--cycle;

% Polygon only rotated, not scaled
fill p0 rotated angle Pr0 withcolor .9white;
draw p0 rotated angle Pr0;

for i=1 upto 4:
  draw q[i] withcolor red;
  draw r[i] withcolor blue;
endfor

pickup pencircle scaled 2;
% Draw Polygons
for i=0 upto 1:
  fill p[i] withcolor .9white;
  draw p[i];
endfor

draw (0,0)--z0 withcolor red;
draw ((0,0)--z0) transformed T withcolor blue;

% Draw Arcs
pickup pencircle scaled .5;
for i=1 upto 4:
  draw c[i] withcolor red;
  draw c[i] scaled abs(Pr0)
            cutafter  ((0, hoehe-.05u)--(breite, hoehe-.05u))
            cutbefore ((breite-.05u, 0)--(breite-.05u, hoehe))
    withcolor blue;
endfor

pic0:=\thelabel.rt(btex \fett T := identity zscaled (2, 1.4)
                   etex, (3.8, 2) transformed t);

unfill bbox pic0;
draw pic0;
pickup pencircle scaled .5;
draw bbox pic0;
```

```
  % Show components of transform T
  Pr1:=(7.5, 7);
  p99:=Pr1--(Pr1+(0, -3.6))--(Pr1+(2, -3.6))--(Pr1+(2, 0))--cycle;
  p99:=p99 shifted (0, .3);
  unfill p99 transformed t;
  draw p99 transformed t;
  label.rt ("Tx="&decimal(xpart  T), Pr1 transformed t); Pr1:=Pr1-(0, .6);
  label.rt ("Ty="&decimal(ypart  T), Pr1 transformed t); Pr1:=Pr1-(0, .6);
  label.rt("Txx="&decimal(xxpart T), Pr1 transformed t); Pr1:=Pr1-(0, .6);
  label.rt("Txy="&decimal(xypart T), Pr1 transformed t); Pr1:=Pr1-(0, .6);
  label.rt("Tyx="&decimal(yxpart T), Pr1 transformed t); Pr1:=Pr1-(0, .6);
  label.rt("Tyy="&decimal(yypart T), Pr1 transformed t);

  pic1:=\thelabel.urt(btex $\mathcal{P}_0$ etex, z0);
  unfill bbox pic1;
  draw pic1;
  draw_point(z0);

  pic2:=\thelabel.urt(btex $\mathcal{P}_0'$ etex, z0 transformed T);
  unfill bbox pic2;
  draw pic2;
  draw_point(z0 transformed T);

  label(btex $\mathcal{F}$ etex,  center p0 shifted (0,.1u));
  label(btex $\mathcal{F}'$ etex, center p1 shifted (0,u));

endfig;

end
```

21 Transform

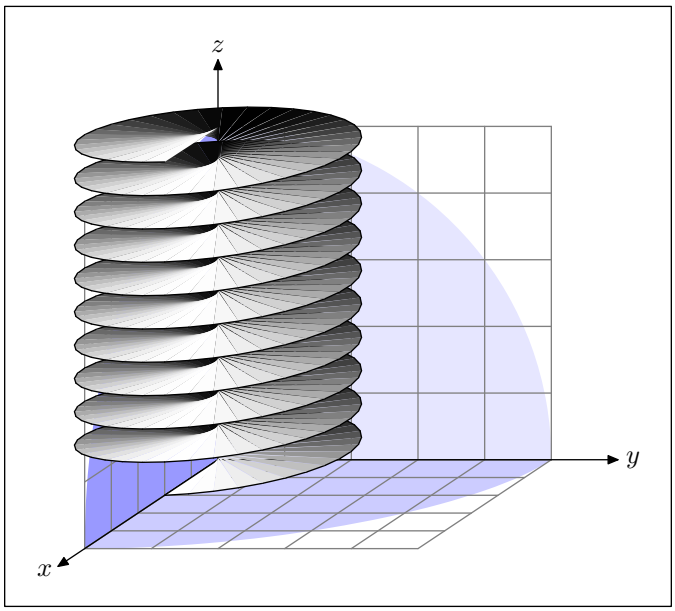



```
u:=25;                    % 25 = 25bp = 25 PostScript points = 30/72 in
wi:=10;                   % width  in units u
he:=8;                    % height in units u
hoehe:=he*u;              % height
breite:=wi*u;             % width

R:=3.7;                   % maximum radius in units u
n:=17;                    % number of edges
phi:=360/(n*10);
phi:=360/(n*5);           % rotation angle

def draw_point(expr P, col) =
  unfill fullcircle scaled 1.5mm shifted P withcolor white;
  draw   fullcircle scaled 1.5mm shifted P withcolor col;
enddef;

transform t, Rot, T;
path q, p[];
pair P[];

Rot:=identity rotated(360/n);
```

```
% ----- Calculations in mathematical coordinates ----------------------------------
P1=(.85, 0);
P2=(1, 0);
P3=P1 transformed Rot;
p0:=P1{1, 2}..{3, -1}P2--P3;                        % p0: first path element
for i=1 upto n-1:                                   % get remaining path elements
  p[i]:=p[i-1] transformed Rot;                      % p[1],..., p[n-1]
endfor                                              % by rotating p0
%
% path (n path elements)
%
p17:=p0 for i=1 upto n-1: &p[i] endfor ..cycle;   % put path elements together

q:=((0, 0)--(1,0)) rotated phi;

P4=p0 intersectionpoint q;
% ----- End of calculations in mathematical coordinates ----------------------------


%
% transform t maps mathematical coordinates on MetaPost coordinates
%
t:=identity scaled (R*u) shifted .5(breite, hoehe);


beginfig(1)    % ========================= figure 1 ================================
  draw (0, 0)--(breite, 0)--(breite, hoehe)--(0, hoehe)--cycle;
  T:=identity;
  for i=0 upto 49:
    p98:=p17 transformed T transformed t;
    if i mod 2 = 0:
      fill p98;
    else:
      unfill p98;
    fi
    draw p98;
    T:=T zscaled P4;
  endfor
endfig;        % ====================== end of figure 1 ============================


beginfig(2)    % ========================= figure 2 ================================
  %
  % Box (in mathematical coordinates)
  %
  path kasten;
  kasten=(.65, -.15)--(1.15, -.15)--(1.15, .5)--(.65, .5)--cycle;
  z98=point 3.5 of kasten transformed t;
  fill kasten transformed t withcolor .9white;
  draw kasten transformed t;

  draw (0, 0)--(breite, 0)--(breite, hoehe)--(0, hoehe)--cycle;

  draw p17 transformed t;
  draw p17 zscaled P4 transformed t withcolor red;
  draw q rotated -phi transformed t;
  draw q transformed t withcolor red;

  pickup pencircle scaled 2;
  draw p0 transformed t;
```

```
  label.rt(btex $p_{17}$ etex, point 3 of p17 transformed t);

  pickup pencircle scaled .5;
  draw_point((0,0) transformed t, black);
  label.bot(btex (0,0) etex, (0,0) transformed t);

  draw_point((1,0) transformed t, black);
  label.bot(btex (1,0) etex, (1,0) transformed t);

  draw_point(P4 transformed t, red);
  label.urt(btex $P_4$ etex, P4 transformed t);

  %
  % renew definition of transform t in order ============
  % to draw the enlarged kasten at the left     ============
  %
  t:=identity scaled (2.3R*u) shifted (-.35breite-1.8u, .5hoehe-.4u);

  kasten:=kasten transformed t;
  fill kasten withcolor .9white;
  draw kasten;
  z99=point 1.5 of kasten;

  draw p0     transformed t;
  draw p1     transformed t cutafter  kasten;
  draw p[n-1] transformed t cutbefore kasten;

  draw p0     zscaled P4 transformed t withcolor red;
  draw p1     zscaled P4 transformed t cutafter  kasten withcolor red;
  draw p[n-1] zscaled P4 transformed t cutbefore kasten withcolor red;

  draw q rotated -phi transformed t cutbefore kasten;
  draw q transformed t cutbefore kasten withcolor red;

  pickup pencircle scaled 2;
  draw p0 transformed t;
  label.rt(btex $p_{17}$ etex, point 3 of p17 transformed t);

  pickup pencircle scaled .5;

  draw_point((1,0) transformed t, black);
  label.bot(btex (1,0) etex, (1,0) transformed t);

  draw_point(P4 transformed t, red);
  label.urt(btex $P_4$ etex, P4 transformed t);

  pickup pencircle scaled 1;
  drawarrow z98--z99;
  pickup pencircle scaled .5;
  draw_point(z98, black);


endfig;        % ====================== end of figure 2 ============================

end
```

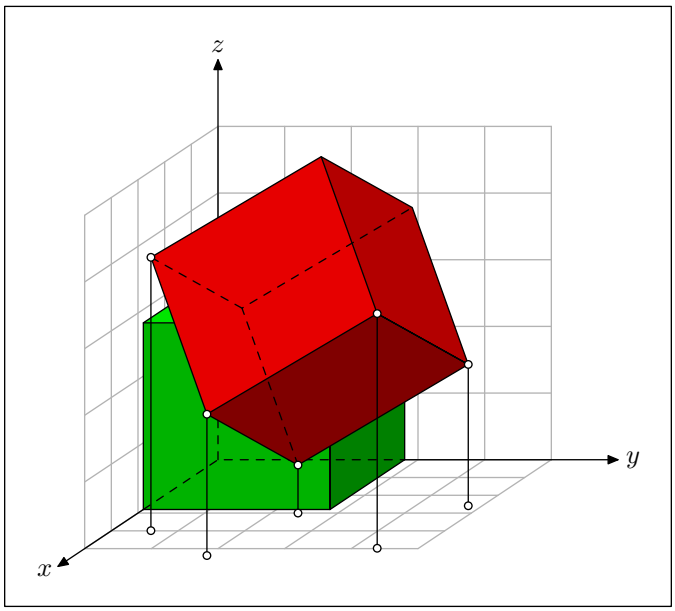
22 Functions




```
u:=25;                          % 25 = 25bp = 25 PostScript points = 25/72 in
wi:=10;                         % width  in units u
he:=9;                          % height in units u
hoehe:=he*u;                    % height
breite:=wi*u;                   % width

transform t, Txy;
pair P[]; path p[];
color viertelskreis_xy[], viertelskreis_yz[], viertelskreis_zx[], spirale[], sp;;

P0=(3.2, 2.2)*u;                % origin in MetaPost coordinates (bp)
P1=(-.6, -.4)/1.5;              % x axis in mathematical coordinates

t:=identity                     % t: maps mathematical 2D coordinates
     scaled u                   %    onto MetaPost coordinates (bp)
       shifted P0;
%
% Txy: maps 3D coordinates x,y onto
%      MetaPost coordinates
%
% Txy is determined by  3  e q u a t i o n s  describing
% how (1,0), (0,1), and (0,0) are mapped
%
   P1=(1,0)transformed Txy;     % Txy: (1,0) --> P1
(1,0)=(0,1)transformed Txy;     %      (0,1) --> (1,0)
(0,0)=(0,0)transformed Txy;     %      (0,0) --> (0,0)
Txy:=Txy transformed t;         %      mathematical 2D coordinates --> MetaPost coordinates

vardef getPixel(expr SpaceVector) =             % returns MetaPost coordinates (bp)
  % SpaceVector: type ``color''                 % of spatial projection of
  (redpart SpaceVector, greenpart SpaceVector)  % 3D point
    transformed Txy                             % with coordinates ``SpaceVector''
      shifted (0, u*bluepart SpaceVector)
enddef;

beginfig(1)
  % --- Abuse of Type <color> for Spatial Coordinates (`viertelskreis' and `spirale') ---
  for i=0 upto 9:
```

```
    viertelskreis_xy[i]:=(cosd 10i, sind 10i, 0);
    viertelskreis_yz[i]:=(0,        cosd 10i, sind 10i);
    viertelskreis_zx[i]:=(sind 10i, 0,        cosd 10i);
  endfor
  for i=0 upto 360:
    spirale[i]:=(2cosd 10i, 2sind 10i, i/72);
  endfor

  % --- Fill halfcircles ---
  fill getPixel(5(viertelskreis_xy0))
    for i=1 upto 9: ..getPixel(5(viertelskreis_xy[i])) endfor
    --getPixel((0,0,0))--cycle
    withcolor (.8, .8, 1);
  fill getPixel(5(viertelskreis_yz0))
    for i=1 upto 9: ..getPixel(5(viertelskreis_yz[i])) endfor
    --getPixel((0,0,0))--cycle
    withcolor (.9, .9, 1) ;
  fill getPixel(5(viertelskreis_zx0))
    for i=1 upto 9: ..getPixel(5(viertelskreis_zx[i])) endfor
    --getPixel((0,0,0))--cycle
    withcolor (.6, .6, 1);

  % --- Grid ---
    color gr;
    gr=.5white;
    for i=0 upto 5:
      draw getPixel((i,0,0))--getPixel((i,5,0)) withcolor gr;
      draw getPixel((0,i,0))--getPixel((5,i,0)) withcolor gr;
      draw getPixel((0,i,0))--getPixel((0,i,5)) withcolor gr;
      draw getPixel((0,0,i))--getPixel((0,5,i)) withcolor gr;
      draw getPixel((0,0,i))--getPixel((5,0,i)) withcolor gr;
      draw getPixel((i,0,0))--getPixel((i,0,5)) withcolor gr;
     endfor
  % --- End Grid ---

  % --- Frame ---
  draw (0, 0)--(breite, 0)--(breite, hoehe)--(0, hoehe)--cycle;

  % --- Axes ---
  drawarrow getPixel((0, 0, 0))--getPixel((6, 0, 0));
  label.llft(btex $x$ etex, getPixel((6, 0, 0))+(0, 1mm));
  drawarrow getPixel((0, 0, 0))--getPixel((0, 6, 0));
  label.rt  (btex $y$ etex, getPixel((0, 6, 0)));
  drawarrow getPixel((0, 0, 0))--getPixel((0, 0, 6));
  label.top (btex $z$ etex, getPixel((0, 0, 6)));

  % --- Draw spiral ---
  for i=0 upto 359:
    sp:=(1+cosd(10i))*.5white;
    fill getPixel((0, 0, bluepart spirale[i]))--getPixel(spirale[i])
      --getPixel(spirale[i+1])--cycle withcolor sp;
    if i=0:
      draw getPixel(spirale[i])--getPixel((0, 0, 0)) ;
    fi
    draw getPixel(spirale[i])--getPixel(spirale[i+1]) ;
  endfor
endfig;

end
```

23 ForSuffixes



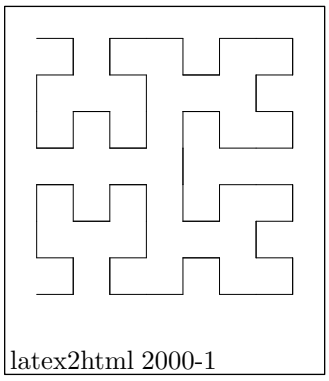

```
u:=25;                          % 25 = 25bp = 25 PostScript points = 30/72 in
wi:=10;                         % width  in units u
he:=9;                          % height in units u
hoehe:=he*u;                    % height
breite:=wi*u;                   % width
transform t, Txy;
pair P[];
path p[];
color wuerfel[], versch;        % 3D vectors: MetaPost type ‘‘color’’

rotX:=45;                       % angle of rotation around the x axis
rotY:=45;                       % angle of rotation around the y axis
rotZ:=45;                       % angle of rotation around the z axis
versch:=(1.6, 1, 2.7);          % translation (in mathematical units)

P0=(3.2, 2.2)*u;                % origin in MetaPost coordinates (bp)
P1=(-.6, -.4)/1.5;              % x axis in mathematical coordinates

t:=identity                                             %   t: maps mathematical 2D coordinates
     scaled u                                           %      onto MetaPost coordinates (bp)
       shifted P0;

Txy:=identity                                           % Txy: maps 3D coordinates x,y onto
       reflectedabout((0,0), (1,1))                     %      MetaPost coordinates
         yscaled ypart P1
   slanted (xpart P1/ypart P1)
     transformed t;

vardef dreheX(expr SpaceVector, winkel) =               % rotation of 3D vector
  pair yz;                                              % ‘‘SpaceVector’’ around
  yz:=(greenpart SpaceVector, bluepart SpaceVector);    % the x axis by the
  yz:=yz rotated winkel;                                % angle ‘‘winkel’’
  (redpart SpaceVector, xpart yz, ypart yz)
enddef;

vardef dreheY(expr SpaceVector, winkel) =               % rotation around the y axis
  pair zx;
  zx:=(bluepart SpaceVector, redpart SpaceVector);
```

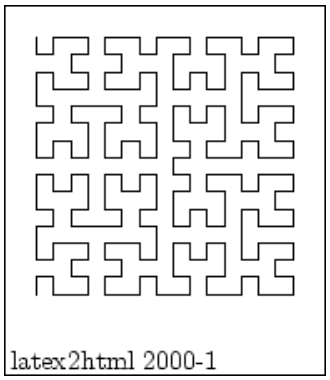
```
  zx:=zx rotated winkel;
  (ypart zx, greenpart SpaceVector, xpart zx)
enddef;

vardef dreheZ(expr SpaceVector, winkel) =              % rotation around the z axis
  pair xy;
  xy:=(redpart SpaceVector, greenpart SpaceVector);
  xy:=xy rotated winkel;
  (xpart xy, ypart xy, bluepart SpaceVector)
enddef;

vardef getPixel(expr SpaceVector) =                    % returns MetaPost coordinates (bp)
  % SpaceVector: type ‘‘color’’                        % of spatial projection of
  (redpart SpaceVector, greenpart SpaceVector)         % 3D point
    transformed Txy                                    % with coordinates ‘‘SpaceVector’’
      shifted (0, u*bluepart SpaceVector)
enddef;

vardef Zyklus(text t) =                % returns cyclic path formed of the
  forsuffixes $=t: z$-- endfor         % z points with suffixes in argument t
  cycle
enddef;

%
%  The following construction copies the possibilities one has in Java with
%
%  main(String[] args) {
%  k = args.length;
%
vardef Pfad(text t) =                  % returns path formed of the
  k:=0;                                % z points with suffixes in argument t
  forsuffixes $=t: k:=k+1; endfor      % count number of arguments first
  i:=1;
  forsuffixes $=t:                     % problem: the last z must not be followed by --
    if i<k: z$-- else: z$ fi
    hide(i:=i+1)                       % hide: to prevent ‘‘i:=i+1’’ to be written into the path
  endfor                               % no ; after ‘‘)’’ (would be written into path)
enddef;

def lot(expr n) =                      % draws perpendicular line from point wuerfel[n]
  color SpaceVector;                   % to (x,y) plane
  SpaceVector:=wuerfel[n];
  draw getPixel(SpaceVector)--getPixel((redpart SpaceVector,greenpart SpaceVector, 0));
  draw_point(getPixel(SpaceVector), white, black);
  draw_point( getPixel((redpart SpaceVector,greenpart SpaceVector, 0)), white, black);
enddef;

def draw_point(expr P, colInt, colPer) =
  fill fullcircle scaled 1mm shifted P withcolor colInt;
  draw fullcircle scaled 1mm shifted P withcolor colPer;
enddef;

def Lote(text t) =                     % invokes lot(n) for all the suffixes in
  forsuffixes $=t:                     % argument t
    lot($);
  endfor
enddef;

wuerfel0:=(2.8, 0, 0);                     % definition of cube ‘‘wuerfel’’
wuerfel1:=(2.8, 2.8, 0);                   % (array of type color)
```

```
wuerfel2:=(0, 2.8, 0);                          % in mathematical 3D coordinates
wuerfel3:=(0, 0, 0);
wuerfel4:=(2.8, 0, 2.8);
wuerfel5:=(2.8, 2.8, 2.8);
wuerfel6:=(0, 2.8, 2.8);
wuerfel7:=(0, 0, 2.8);


beginfig(1)
  for i=0 upto 7:                               %  z0,...,z7: MetaPost coordinates of
    z[i]=getPixel(wuerfel[i]);                  %  cube in original position
  endfor

  for i=0 upto 7:                               % rotation and translation
    wuerfel[i]:=dreheX(wuerfel[i], rotX);       % of cube
    wuerfel[i]:=dreheY(wuerfel[i], rotY);
    wuerfel[i]:=dreheZ(wuerfel[i], rotZ);
    wuerfel[i]:=wuerfel[i]+versch;
  endfor

  for i=0 upto 7:                               %  z100,...,z107: MetaPost coordinates of
    z[i+100]=getPixel(wuerfel[i]);              %  cube after rotation
  endfor

  % --- Grid ---
    for i=0 upto 5:
      draw getPixel((i,0,0))--getPixel((i,5,0)) withcolor .7white;
      draw getPixel((0,i,0))--getPixel((5,i,0)) withcolor .7white;
      draw getPixel((0,i,0))--getPixel((0,i,5)) withcolor .7white;
      draw getPixel((0,0,i))--getPixel((0,5,i)) withcolor .7white;
      draw getPixel((0,0,i))--getPixel((5,0,i)) withcolor .7white;
      draw getPixel((i,0,0))--getPixel((i,0,5)) withcolor .7white;
     endfor
  % --- End Grid ---

  % --- Frame ---
  draw (0, 0)--(breite, 0)--(breite, hoehe)--(0, hoehe)--cycle;

  % --- Axes ---
  drawarrow getPixel((0, 0, 0))--getPixel((6, 0, 0));
  label.llft(btex $x$ etex, getPixel((6, 0, 0))+(0, 1mm));
  drawarrow getPixel((0, 0, 0))--getPixel((0, 6, 0));
  label.rt  (btex $y$ etex, getPixel((0, 6, 0)));
  drawarrow getPixel((0, 0, 0))--getPixel((0, 0, 6));
  label.top (btex $z$ etex, getPixel((0, 0, 6)));

  % --- Cube in original position ---
  fill Zyklus(0, 1, 5, 4) withcolor .7green;
  fill Zyklus(4, 5, 6, 7) withcolor .9green;
  fill Zyklus(1, 2, 6, 5) withcolor .5green;
  draw Pfad(0, 3, 2)      dashed evenly;
  draw Pfad(3, 7)         dashed evenly;
  draw Zyklus(1, 2, 6, 7, 4, 0) ;
  draw Pfad(4, 5, 1, 5, 6);

  % --- Cube after rotation ---
  fill Zyklus(104, 105, 106, 107) withcolor .9red;
  fill Zyklus(101, 102, 106, 105) withcolor .7red;
  fill Zyklus(100, 101, 105, 104) withcolor .5red;
  draw Pfad(100, 103, 102) dashed evenly;
```

```
  draw Pfad(103, 107)      dashed evenly;
  draw Pfad(104, 105, 101, 105, 106);
  draw Zyklus(101, 102, 106, 107, 104, 100);
  Lote(0, 1, 4, 5, 7);                      % suffixes of ‘‘wuerfel’’
endfig;

end
```

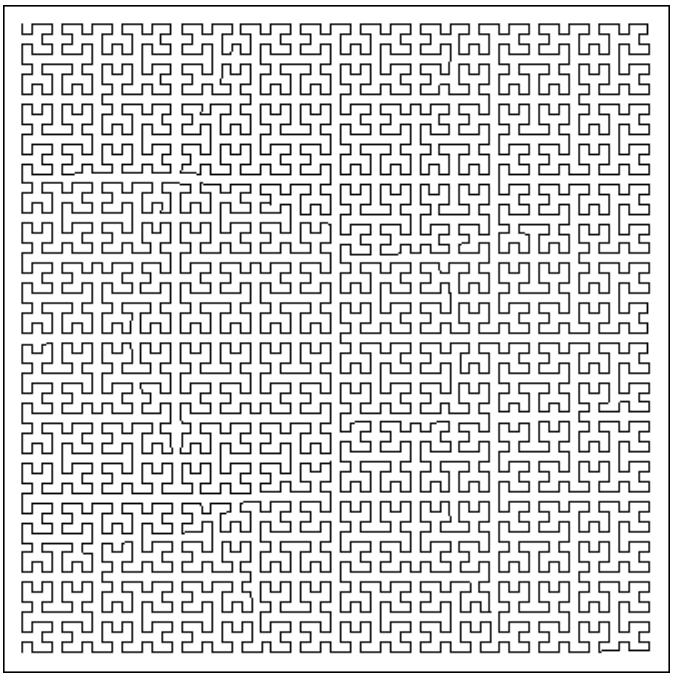
24 Recursiveness






```
u:=12;
wi:=10;                         % width  in units u
he:=11.5;                       % height in units u
hoehe:=he*u;                    % height
breite:=wi*u;                   % width
rd:=u;
bHilbert:=breite-2rd;
P0:=(rd, hoehe-rd);
pair P[];

vardef hilbert(expr s, n) =    %  returns a <picture> h
  pair ur, lr, ref[];
  picture h, hOLD;
  h:=currentpicture;
  clip h to P0--P0--P0--P0--cycle;
  if n>0:
    hOLD:=hilbert(s, n-1);                          % ‘‘hilbert(s, n)’’ envokes itself!
    hOLD:=hOLD rotatedaround(center hOLD, -90);
    clip h to P0--P0--P0--P0--cycle;                % necessary to get rid of old h
    addto h also hOLD;
    ur:=urcorner h; lr:=lrcorner h;
    ref1:=(xpart P0, ypart lr - .5s);               % first point of axis of reflection
    ref2:=ref1 shifted (s, 0);                      % second point of axis of reflection
    addto h doublepath ur--(ur shifted (s, 0));
    addto h doublepath (lr shifted (s, 0))--(lr shifted (s, -s));
    w:=xpart(ur-P0);
    addto h also hOLD rotatedaround(center hOLD, 90) shifted (s+w, 0);
    addto h also (h reflectedabout(ref1, ref2));
  fi
  h
enddef;

beginfig(1)
  draw (0, 0)--(breite, 0)--(breite, hoehe)--(0, hoehe)--cycle;
  draw hilbert(bHilbert/7, 3);                      % second argument 3: depth 3
  label.urt("latex2html 2000-1", (0,0));
endfig;

beginfig(2)
  draw (0, 0)--(breite, 0)--(breite, hoehe)--(0, hoehe)--cycle;
  draw hilbert(bHilbert/15, 4);                     % second argument 4: depth 4
  label.urt("latex2html 2000-1", (0,0));
endfig;

end
```

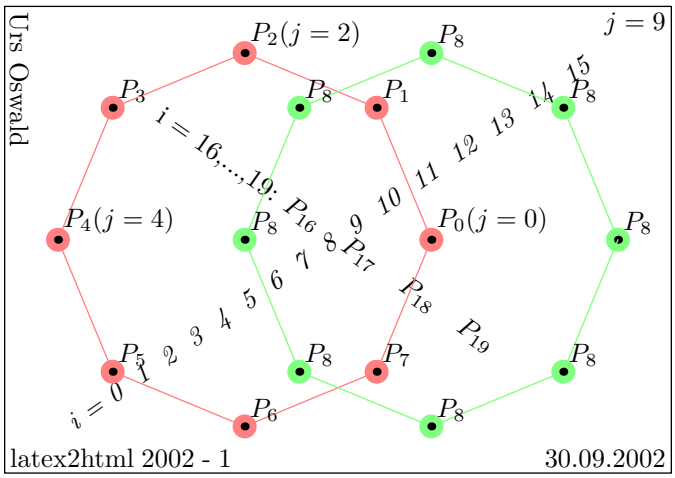
25 RecursivePath



```
%
%  /home/osurs/latex/metapost/TutorialAugsburg/tutorial/RecursivePath/RecursivePath.mp
%  16.09.02
%

u:=25;                    % 25 = 25bp = 25 PostP102cript points = 25/72 in
wi:=10;                   % width  in units u
he:=10;                   % height in units u
hoehe:=he*u;              % height
breite:=wi*u;             % width
path p[];
pair P[], versch, vs;

P0:=(6.875, hoehe-6.875);     % starting point top left

vardef hilbertPath(expr s, n) =
  %
  %  recursively calculates and returns path ‘‘hilb’’
  %  of segment length s and depth n
  %
  pair versch;
  path hilb, hilbOLD;
  versch:=(s, 0);
  % --- Calculation of path ‘‘hilb’’ ---
  if n=0:
    hilb:=P0;
  else:
    hilbOLD:=hilbertPath(s, n-1);
    p100:=hilbOLD reflectedabout(P0, P0 shifted (1, -1));
    P100:=point length(p100) of p100;
    vs:=P100-P0;
    p101:=hilbOLD shifted (vs+versch);
    P101:=point 0 of p101;
    hilb:=p100&P100--P101&p101;
    P102:=P101 shifted ((vs+.5versch) rotated -90);
    p102:=reverse hilb reflectedabout(P102, P102 shifted (1,0));
```

```
    hilb:=hilb&point length(hilb) of hilb--point 0 of p102&p102;
  fi
  % --- End: Calculation of path ‘‘hilb’’ ---
  hilb         % return path ‘‘hilb’’
enddef;

beginfig(1)
  draw (0, 0)--(breite, 0)--(breite, hoehe)--(0, hoehe)--cycle;

  draw hilbertPath(3.72, 6);  % Draw Hilbert curve of segment length
                              % 3.72 PostScript points and depth 6

  % draw hilbertPath(2, 7);   % maximum possible depth
                              % with main memory size=1000001
endfig;

end
```

26 ifthenInLabel




```
verbatimtex
%&latex
\documentclass{article}
\newcounter{i} \setcounter{i}{0}   % LaTeX counter i=0
\newcounter{j} \setcounter{j}{0}   % LaTeX counter j=0
\usepackage{ifthen}

\begin{document}
etex

breite=250; hoehe=175;
R:=.4hoehe;
v:=35;

beginfig(1)
  %
  % --- Left polygon ---
  %
  for k=0 upto 7: z[k]=(.5breite+R*cosd 360/8k-v, .5hoehe+R*sind 360/8k); endfor
  %
  % --- Frame ---
  %
  draw (0,0)--(breite, 0)--(breite, hoehe)--(0, hoehe)--cycle;
  %
  % --- LaTex counter i=0,...,15
  %
  label(btex
          \itshape
  i =
  \whiledo{\value{i}<16}{%
    \thei\ \stepcounter{i}
  }%
        etex rotated angle(breite, hoehe), .5(breite, hoehe));
  %
  % --- LaTex counter i=16,...,19
  %
  label(btex
          i = 16,...,19:
  \whiledo{\value{i}<20}{%
    $P_{\thei}$\hspace{.5em} \stepcounter{i}
  }%
        etex rotated -angle(breite, hoehe), .5(breite, hoehe));
```

```
  %
  % --- Red polygon and knots ---
  %
  for k=0 upto 7:
    draw z[k]--z[(k+1)mod 8]              withcolor (1, .5, .5);
    fill fullcircle scaled 9 shifted z[k] withcolor (1, .5, .5);
  endfor
  %
  % --- LaTeX counter j=0,...,8 (red circles) ---
  %
  dotlabel.urt(btex $P_{\thej}(j=\thej)$ \stepcounter{j} etex, z[0]);
  dotlabel.urt(btex $P_{\thej}$ \stepcounter{j} etex, z[1]);
  dotlabel.urt(btex $P_{\thej}(j=\thej)$ \stepcounter{j} etex, z[2]);
  dotlabel.urt(btex $P_{\thej}$ \stepcounter{j} etex, z[3]);
  dotlabel.urt(btex $P_{\thej}(j=\thej)$ \stepcounter{j} etex, z[4]);
  dotlabel.urt(btex $P_{\thej}$ \stepcounter{j} etex, z[5]);
  dotlabel.urt(btex $P_{\thej}$ \stepcounter{j} etex, z[6]);
  dotlabel.urt(btex $P_{\thej}$ \stepcounter{j} etex, z[7]);
  %
  % --- LaTeX counter j is not stepped up! Keeps value j=8! (green circles) ---
  %
  for k=0 upto 7:
    draw (z[k]--z[(k+1)mod 8])shifted (2v, 0)       withcolor (.5, 1, .5);
    fill fullcircle scaled 9 shifted (z[k]+(2v, 0)) withcolor (.5, 1, .5);
    dotlabel.urt(btex $P_{\thej}$ \stepcounter{j} etex, z[k]shifted (2v, 0));
  endfor
  %
  % --- Final value of j:
  %
  label.llft(btex $j=\thej$ etex, (breite, hoehe));

  label.urt("latex2html 2002 - 1", (0,0)); label.ulft("30.09.2002", (breite, 0));
  label.lrt(btex Urs Oswald etex rotated -90, (0, hoehe));
endfig;

end
```

27 NewOperators

Binary operators defined with `tertiarydef`, `secondarydef`, and `primarydef` expect the right argument to be of *tertiary*, *secondary*, and *primary* precedence level, respectively, whereas the left argument is expected to be one level off. But see the `primarydef` examples!

```
osurs@linux:~/latex/metapost> mpost
This is MetaPost, Version 0.641 (Web2C 7.3.7)
**\relax
```

```
*tertiarydef p terdef q = 2p+3q enddef;

*show 6 terdef 35;
>> 117
*show 6 terdef 4*8+3;
>> 117
*show 2+2*2 terdef 35;
>> 117
*show 6 terdef 4*8+3=117;
>> true
*show 6 terdef 4*8+3=118;
>> false
*show 6=2+2*2 terdef 35;
>> 2
>> true
! Not implemented: (known numeric)*(boolean).
```

As this example shows, a binary operator `terdef` defined with `tertiarydef` expects a *tertiary* right argument, whereas the left argument can be of *expression* type.

```
*secondarydef p secdef q = 2p+3q enddef;

*show 6 secdef 35;
>> 117
*show 2*3 secdef 5*7;
>> 117
*show 6 secdef 4*8+3;
>> 111
*show (6 secdef 4*8)+3;
>> 111
*show 2+2*2 secdef 35;
>> 117
*show 7=2+2*2 secdef 35;
>> false
```

As this example shows, a binary operator `secdef` defined with `secondarydef` expects a *secondary* right argument, whereas the left argument can be of *tertiary* type.

```
*primarydef p primdef q = 2p+3q enddef;

*show 6 primdef 35;
>> 117
*show 6 primdef 5*7;
>> 117
*show 6 primdef 4*8+3;
>> 111
*show (6 primdef 4*8)+3;
>> 111
```

This binary operator `primdef` defined with `primarydef` also expects a *secondary* right argument!

```
*primarydef p primdef\_k q = (2p+3q) enddef;

*show 6 primdef\_k 5*7;
>> 189
*show (6 primdef\_k 5)*7;
>> 189
*show 2*3 primdef\_k 35;
>> 117
*show 100+2*3 primdef\_k 35;
>> 217
*show 100+(2*3 primdef\_k 35);
>> 217
*end
```

After enclosing the value to be returned in parentheses (or BEGINGROUP ... ENDGROUP) this binary operator `primdef_k` defined with `primarydef` expects a *primary* right argument, whereas the left argument can be of *secondary* type.