a vector graphics language for technical drawing

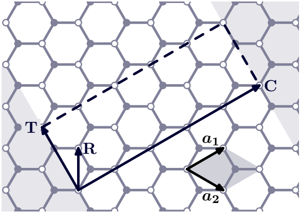
```
import graph;

size(140,80,
     IgnoreAspect);

picture logo(pair s=0, pen q)
{
  picture pic;
  pen p=linewidth(2)+fontsize(24)+q;
  real a=-0.4;  real b=0.95; real y=5;
  path A=(a,0){dir(10)}::{dir(89.5)}(0,3y/2);
  draw(pic,A,p);
  draw(pic,(0,-y){dir(88.3)}::{dir(20)}(b,0),p);
  real c=0.5*a; pair z=(0,2.5);
  label(pic,"{\it symptote}",z,0.25*E+0.169S,p);
  pair w=(0,1.7);
  draw(pic,intersectionpoint(A,w-1--w)--w,p);
  axes(pic,p);
  return shift(s)*pic;
}

pair z=(-0.015,0.08);
for(int x=0; x < 10; ++x)
  add(logo(0.1*x*z,gray(0.04*x)));

add(logo(red));
shipout(format="pdf");
```

25.01.2008

Ralf Hambach

What is it ?

- like MetaPost, coordinate based
- vector graphics language (high level) → scripting and gui
- labels in LaTeX
- Output: eps, pdf, any ImageMagick format



Features

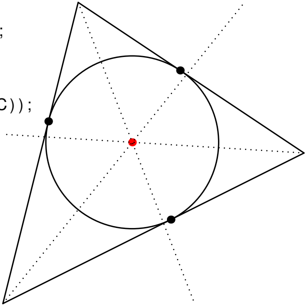
- integration with LaTeX
- like MetaPost: mathematically oriented →exact and parametrized
- C++-like programming syntax
- 3D vectors and graphs
- packages for additional purposes
- platform independent (UNIX, MacOS, Win)

Drawbacks

- time-consuming, but perfect
- under development, however stable
- XFig like user-interface (rudimentary)

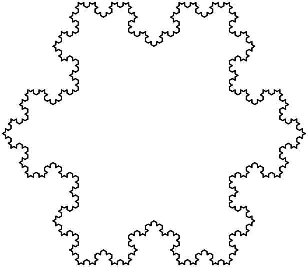
Geometrical Figures

```
size(4cm,0);

import math;

pair A=(0,0), B=(1,.5), C=(.25,1);

pair project(pair pt, pair A, pair B){
return extension(\
  pt,pt-dir(90+degrees(A--B,false)),A,B);
}
pair icenter(pair A, pair B, pair C){
return extension(\
  A,A+dir(A--B,A--C), B, B+dir(B--A,B--C));
}
draw(A--B--C--cycle);

pair ins=icenter(A,B,C);
pair iAB=project(ins,A,B);
pair iAC=project(ins,A,C);
pair iBC=project(ins,B,C);

dot(ins, red);
dot(iAB^^iAC^^iBC);
drawline(A, ins, dotted);
drawline(B, ins, dotted);
drawline(C, ins, dotted);
draw(shift(ins)*scale(abs(ins-iAB))\
               *unitcircle);

shipout(format="pdf");
```



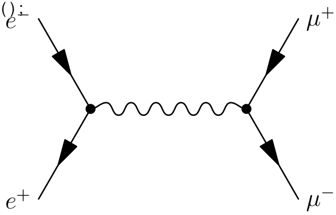
Programming

```
size(0,0);
real u=2cm;
void koch(pair A, pair B, int n)
{
  pair C;
  C =rotate(120, point(A--B,1/3))*A;
  if (n>0)
    {
      koch( A,point(A--B,1/3),   n-1);
      koch(   point(A--B,1/3), C,n-1);
      koch( C,point(A--B,2/3),   n-1);
      koch(   point(A--B,2/3), B,n-1);
    }
  else draw(A--point(A--B,1/3)
            --C--point(A--B,2/3)--B);
}

pair z0=(u,0);
pair z1=rotate(120)*z0;
pair z2=rotate(120)*z1;
koch( z0, z1, 3 );
koch( z1, z2, 3 );
koch( z2, z0, 3 );

shipout(format="pdf");
```

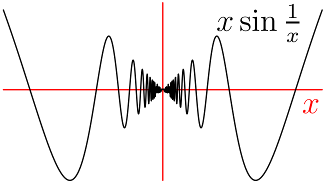


Feynman

```
import feynman;

currentpen = linewidth(0.8); fmdefaults();

real L = 50;
pair zl = (−0.75∗L,0);
pair zr = (+0.75∗L,0);
pair xu = zl + L∗dir(+120);
pair xl = zl + L∗dir(−120);
pair yu = zr + L∗dir(+60);
pair yl = zr + L∗dir(−60);

drawFermion(xu—zl);
drawFermion(zl—xl);
drawPhoton(zl—zr);
drawFermion(yu—zr);
drawFermion(zr—yl);
drawVertex(zl);
drawVertex(zr);

label(”$eˆ−$”, xu, left);
label(”$eˆ+$”, xl, left);
label(”$\muˆ+$”, yu, right);
label(”$\muˆ−$”, yl, right);

shipout(format=”pdf”);
```



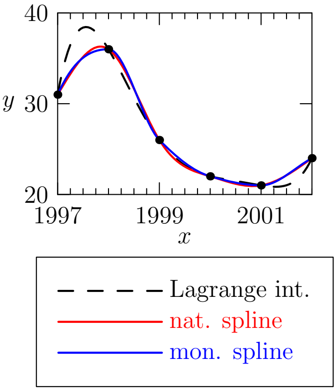
Graphs 1D



```
import graph;
size(120,0);

real f(real x){
return (x != 0.0)? x*sin(1.0/x) :0.0;
}
pair F(real x){
return (x,f(x));
}

xaxis(”$x$”,red);
yaxis(red);
draw(graph(f,−1.2/pi,1.2/pi,1000));
label(”$x\sin\frac{1}{x}$”,F(1.1/pi),NW);

shipout(format=”pdf”);
```

Scientific Graphs

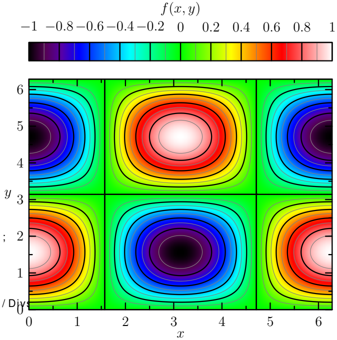
```
import graph;
import interpolate;

size(2cm,4cm,IgnoreAspect);

real a=1997, b=2002;
int n=5;
real[] xpt=a+sequence(n+1)*(b-a)/n;
real[] ypt={31,36,26,22,21,24};
horner h=diffdiv(xpt,ypt);
fhorner L=fhorner(h);

scale(false,true);

pen p=linewidth(1);

draw(graph(L,a,b),dashed+black+p,
     "Lagrange␣int.");
draw(graph(xpt,ypt,Hermite(natural)),
     red+p,"nat.␣spline");
draw(graph(xpt,ypt,Hermite(monotonic)),
     blue+p,"mon.␣spline");
xaxis("$x$",BottomTop,
      LeftTicks(Step=2,step=0.25));
yaxis("$y$",LeftRight,
      RightTicks(Step=10));
dot(xpt,ypt,4bp+0.7black);

attach(legend(),point(10S),30S);

shipout(format="pdf");
```



Graphs 2D

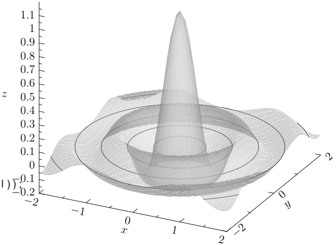
```
import graph;
import palette;
import contour;

size(10cm,10cm,IgnoreAspect);

pair a=(0,0);
pair b=(2pi,2pi);
```

```
real f(real x, real y) {return cos(x)*sin(y);}

int N=200;
int Divs=10;
int divs=2;

defaultpen(1bp);
pen Tickpen=black;
pen tickpen=gray+0.5*linewidth(currentpen);
pen[] Palette=BWRainbow();

scale(false);

bounds range=image(f,Automatic,a,b,N,Palette);

// Major contours
real[] Cvals;
Cvals=sequence(Divs+1)/Divs*(range.max-range.min)+range.min;
draw(contour(f,a,b,Cvals,N,operator --),Tickpen);

// Minor contours
real[] cvals;
real[] sumarr=sequence(1,divs-1)/divs*(range.max-range.min)/Divs
for (int ival=0; ival < Cvals.length-1; ++ival)
  cvals.append(Cvals[ival]+sumarr);
draw(contour(f,a,b,cvals,N,operator --),tickpen);

xaxis("$x$",BottomTop,LeftTicks,Above);
yaxis("$y$",LeftRight,RightTicks,Above);

palette("$f(x,y)$",range,point(NW)+(0,0.5),point(NE)+(0,1),Top,Palette,
        PaletteTicks(N=Divs,n=divs,Tickpen,tickpen));

shipout(format="pdf");
```



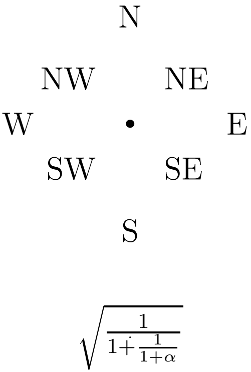
3D Objects

```
import graph3;
import contour;

size(12cm,0);

real sinc(pair z) {
  real r=2pi*abs(z);
  return r != 0 ? sin(r)/r : 1;
}

bbox3 b=limits((—2,—2,—0.2),(2,2,1.2));
currentprojection=orthographic(1,—2,1);
currentlight=(1,—1,0.5);

aspect(b,1,1,1);

xaxis(rotate(X)*"$x$",b,RightTicks(rotate(X)*Label));
yaxis(rotate(Y)*"$y$",b.X(),b.XY(),LeftTicks(rotate(Y)*Label));
zaxis("$z$",b,RightTicks());

layer();

draw(lift(sinc,contour(sinc,(—2,—2),(2,2),new real[] {0})));
add(surface(sinc,xypart(b.O()),xypart(b.XY()),50,lightgray+opacity(0.5)));

shipout(format="pdf");
```

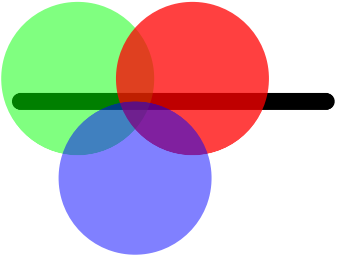


Labelling

```
size(4cm,0);

pair O=0;
dot(”N”,O,10N);
draw(”S”,O,10S);
draw(”E”,O,10E);
draw(”W”,O,10W);

draw(”NE”,O,5NE);
draw(”SE”,O,5SE);
draw(”NW”,O,5NW);
draw(”SW”,O,5SW);

draw(”$\sqrt{\frac{1}{1+
␣␣\frac{1}{1+\alpha}}}$”,80S,O);

shipout(format=”pdf”);
```



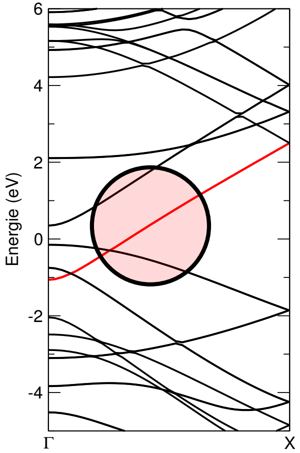
Transparency



```
size(0,150);

draw((-1.5,1)--(2.5,1),linewidth(10));
fill(shift(1.5dir(120))
   *unitcircle ,green+opacity(0.5));
fill(shift(1.5dir(60))
     *unitcircle ,red+opacity(0.75));
fill(unitcircle ,blue+opacity(0.5));

shipout(format="pdf");
```

Importing EPS

```
size(0,0);

label(graphic("CNT_5_0_bandstruktur.eps",
              "width=5cm"));
layer();

 fill(scale(1cm)*unitcircle,
  red+opacity(0.15));
draw(scale(1cm)*unitcircle,
  linewidth(2pt));

shipout(format="pdf");
```



A: Hello Line I

Aim: different modes of running asymptote and its integration into LaTeX.

- Starting from the command line

```
sh> asy
> draw((0,0)--(100,100));
> q
```

- Reading from file (same content as above)

```
sh> asy -V line.asy     # visualization
sh> asy line.asy        # writes ⋆.eps
```

- or even interactively (install: python-tk, python-tkinter)

```
sh> xasy line.asy
```

A: Hello Line II

- Integration into latex (besides `\includefigure`)

```
\documentclass[12pt]{article}
\usepackage{asymptote}
\begin{document}

\begin{asy}
  draw((0,0)--(100,100));
\end{asy}

\end{document}
```

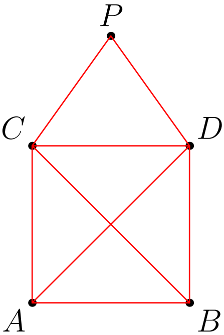
- Then run pdflatex → asy → pdflatex:

```
sh> pdflatex line.tex;
sh> asy      line;
sh> pdflatex line.tex;
```

B: Knecht Ruprecht

Aim: start programming

Use Asymptote to solve the well known riddle...

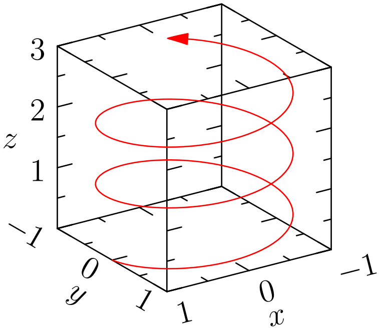


```
size(4cm,0);
pair A=(0,0); ...
draw(P--C--D--P);
dot(Label("$A$"),A,SW);
```

C: 3D Twist

Aim: extended drawings

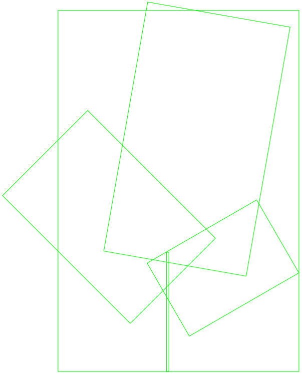
```
import graph3;

currentprojection=...;

// parametrization
real x(real t) {return ...;}
real y(real t) {return ...;}
real z(real t) {return ...;}

// define path
path3 p=graph(x,y,z,<start>,<end>,
                         operator ..);

draw(p);
```



D: Growing Smilies I

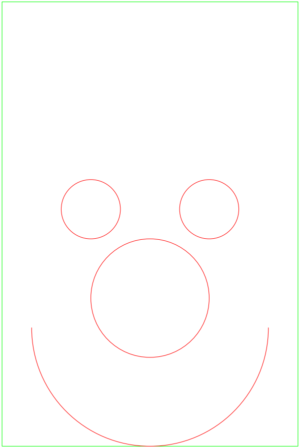
Aim: a growing number of happy faces

```
transform ta=shift(...,...)*
             rotate(...)*
             scale(...,...);

picture smilie,pic_out;
draw(smilie,unitsquare);

add(pic_out,ta*smilie);

//final output
draw(pic);
```



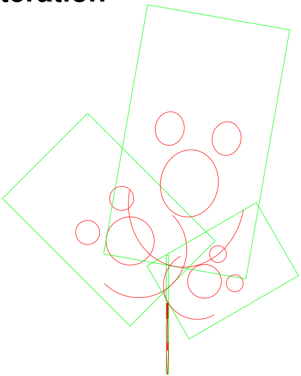
D: Growing Smilies II

Aim: Input



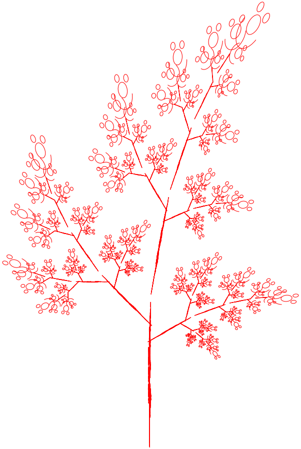
D: Growing Smilies III

Aim: first Iteration



D: Growing Smilies !

Aim: 6. Iteration



B: Knecht Ruprecht

```
size(3cm,0);

pair A=(0,0), B=(1,0), C=(0,1), D=(1,1), P=(0.5,1.7);

dot(Label("$A$"),A,SW);
dot(Label("$B$"),B,SE);
dot(Label("$C$"),C,NW);
dot(Label("$D$"),D,NE);
dot(Label("$P$"),P,N);

draw(A--C--B--D--P--C--D--A--B,red);
//draw(A..C..B..D..P..C..D..A..B,Arrow);
//draw(A--C--B--D--P..C..D..A..B);

shipout(format="pdf");
```

C: 3D Twist

```
import graph3;

size(0,120);

currentprojection=orthographic(4,6,3);

real x(real t) {return cos(2pi*t);}
real y(real t) {return sin(2pi*t);}
real z(real t) {return t;}

defaultpen(overwrite(SuppressQuiet));

path3 p=graph(x,y,z,0,2.7,operator ..);
bbox3 b=autolimits(min(p),max(p));
aspect(b,1,1,1);

xaxis(rotate(X)*"$x$",all=true,b,
        RightTicks(rotate(X)*Label,2,2));
yaxis(rotate(Y)*"$y$",all=true,b,
        RightTicks(rotate(Y)*Label,2,2));
zaxis("$z$",all=true,b,RightTicks);

draw(p,red,Arrow);

shipout(format="pdf");
```

D: Growing Smilies

```
size(10cm,0);

// hand-tuning for nice attractor
transform ta=shift(.45,0)*scale(.01,.33);
transform tb=shift(.3,.2)*rotate(45)*scale(0.5);
transform tc=shift(.37,.45)*rotate(-60)*scale(0.35);
transform td=shift(0.19,0.5)*rotate(-10)*scale(0.6,0.7);


picture smilie,rect;
draw(rect,scale(1,1.5)*unitsquare,green);
//smilie=rect;
draw(smilie,shift(0.3,0.8)*scale(0.1)*unitcircle,red);
draw(smilie,shift(0.7,0.8)*scale(0.1)*unitcircle,red);
draw(smilie,shift(0.5,0.5)*scale(0.2)*unitcircle,red);
draw(smilie,(0.1,0.4)..(0.5,0)..(0.9,0.4),red);

picture Iterate(picture pic_in) {
  picture pic_out;
  add(pic_out,ta*pic_in);
  add(pic_out,tb*pic_in);
  add(pic_out,tc*pic_in);
  add(pic_out,td*pic_in);

  return pic_out;
}


for(int i=0; i<5; ++i) {
  smilie=Iterate(smilie);
}
add(smilie);

shipout(prefix="fractal4",format="pdf");
```

References

- Official Homepage: `http://asymptote.sourceforge.net/`
- Examples (french): `http://piprim.tuxfamily.org/asymptote/index.html`