Министерство образования науки Российской Федерации

САНКТ-ПЕТЕРБУРГСКИЙ
ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО

Институт прикладной математики и механики
Высшая школа прикладной математики и вычислительной физики

А.Н.Баженов

Подготовка иллюстраций для научных и технических публикаций для LaTeX.

Учебное пособие

Санкт-Петербург

2020

УДК 519.682.3

А в т о р :
А.Н. Баженов Подготовка иллюстраций для научных и технических публикаций для LaTeX. – СПб., 2020. – 70 с.

Учебное пособие соответствует образовательному стандарту высшего образования Федерального государственного автономного образовательного учреждения высшего образования «Санкт-Петербургский политехнический университет Петра Великого» по направлению подготовки бакалавров 01.03.02 «Прикладная математика и информатика», по дисциплине «Вычислительные комплексы».

Пособие посвящено подготовке иллюстраций для научных и технических публикаций LaTeXс помощью `MetaPost` — интерпретатора языка программирования META и сходных инструментов.

Материал апробирован в учебных курсах для студентов кафедры «Прикладная математика» Института Прикладной Математики и Механики СПбПУ и аспирантов первого года обучения ФТИ им. А.Ф.Иоффе РАН.

Содержание

Список примеров

Список иллюстраций

Аннотация

В пособии приведены учебные примеры для интерпретатора языка программирования META `METAPOST` [1] и родственных с ним средств подготовки векторных изображений. `METAPOST` содержит большое количество функций, ориентированных на геометрические построения и позволяет использовать средства LaTeX для текста и формул на иллюстрациях. Дополнительное достоинство векторного способа представления иллюстраций — возможность просмотра многими программами, экспортирования в другие форматы и обработки существующими графическими редакторами.

Пособие построено следующим образом. Оно состоит из разделов, в которых последовательно приводятся различные примеры. Для каждого примера приводится текст программы, ключевые моменты и результат работы: иллюстрация.

Все процитированные примеры, не принадлежащие автору, имеют ссылку в Интернет. Как правило, они взяты с ресурса CTAN (The Comprehensive TeX Archive Network) [2], а также texample [46].

Пример единства вычислений и иллюстрирования - чертеж из рукописи Архимеда

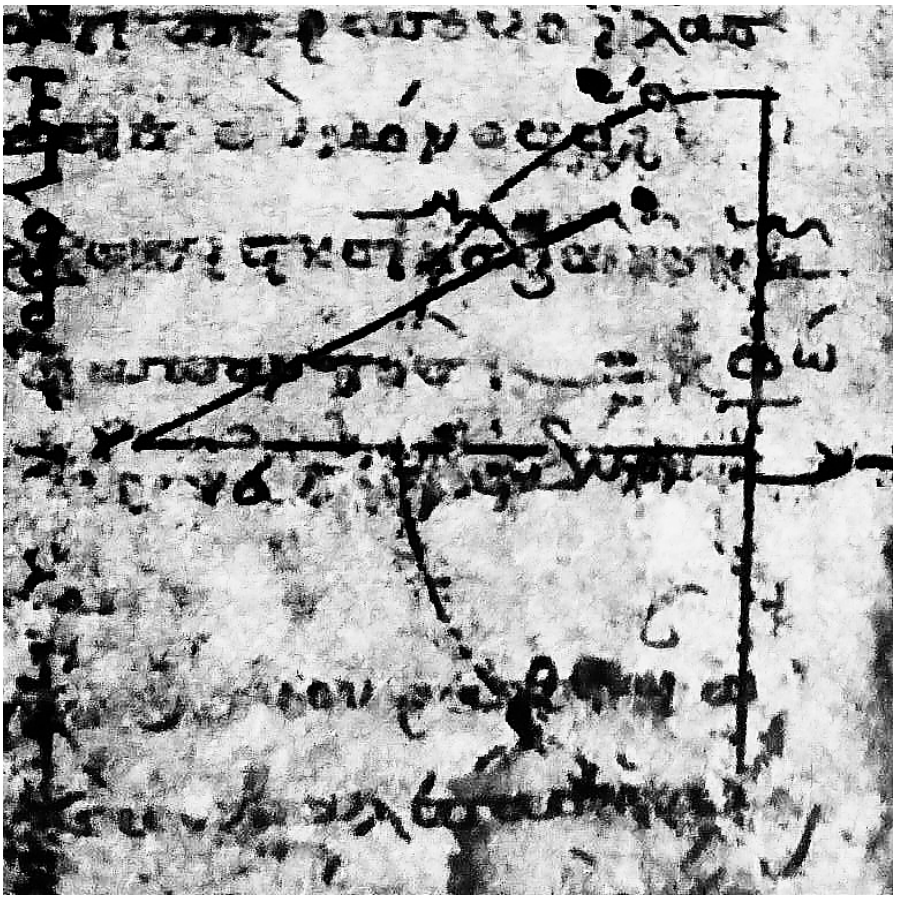

Рис. 1: Чертеж из рукописи Архимеда

1 Введение

Зачем изучать METAPOST ? Какова мотивация использования дополнительного продукта при возможности создания графических файлов огромным количеством способов?

При подготовке учебных и научных материалов автор использует систему LaTeX. В некоторых случаях ее использование является делом вкуса, а в некоторых - обязательно. В основном последний случай относится к математическим изданиям, а в последнее время и к квалифкационым работам. LaTeXмотивирует использование единого стиля подготовки материала, при этом иллюстрации импортиуются как внешние файлы. Ввиду такого разделения возникает проблема единого стиля оформления текста и графики.

Кроме того, если иллюстрируется математический факт, обычно в геометрических образах, весьма желательно, чтобы иллюстрация точно отражала суть вопроса. Желательно, чтобы кривые второго порядка на иллюстрации были именно эллипсами, параболами и гиперболами, нормаль пересекала кривую под прямым углом и т.п. Метки, стрелки, обозначения углов и прочий аннотационый материал также должен быть привязан к геометрическим объектам.

Раньше автор готовил иллюстрации в пакете Matlab, программируя геометрические объекты и создавая рисунки. Выдержать единый стиль с текстом при этом очень непросто. О существовании и возможностях системы METAPOST автор узнал из книги [3] и использовал ее для создания иллюстраций в пособии по интервальному анализу [4]. Часть из них вошла в данное пособие.

Важная черта METAPOST - единство геометрии и аннотирования, манипуляция с элементами аннотирования и фрагментами построений как геометрическим объектами. Привязка аннотаций к ключевым геометрическим местам при этом возникает естественным образом. «MetaPost — это программа для тех, кто может объяснить компьютеру что он хочет. MetaPost чрезвычайна полезен для случаев, когда картинку проще описать логически, нежели образно» [8].

Про METAPOST существует большое количество руководств, пособий и презентаций на английском и русском языках. Некоторые из них приведены в библиографии настоящего пособия. Отмечу курс для школьников при МГУ [5]. Очень поучительная подборка примеров доступна по ссылке [6].

В METAPOST много тонкостей и технических приемов. В связи с наличием доступных материалов, в данном пособии нет систематического изложения структур и методов METAPOST , внимание сконцентрировано на некоторых базовых идеях и примерах, иллюстрирующих применение. Таким образом, цель пособия в основном - пропаганда «честного» подхода к иллюстрации: желательно, чтобы иллюстрация не просто была похожа на обсуждаемый объект, а точно соотвествовала ему.

2 Иллюстрации в METAPOST

Для подготовки иллюстраций в METAPOST нужно подготовить текстовый файл с инструкциями в любом редакторе. Далее с поомощью утилиты mpost создается файл в формате EPS, который можно использовать в документах LaTeX. Есть возможности интеграции этого процесса в процесс обычной одготовки документов. Например, в среде Texstudio [7] есть справочник функций METAPOST , а утилита mpost вызывается из меню или с помощью предопределенной комбинации клавиш.

Одного и того же результата можно достичь по-разному. Автор не является гуру METAPOST , и многое в примерах сделано неоптимально и не самым элегантным способом. Мне представляется более важным, чтобы построение было достоверным, основывалось на математических и физических фактах, определяющих геометрические места.

Пример 2.1 (Привязка меток к объектам на иллюстрациях [1])
Начнем с самого начала и рассмотрим базовый пример интеграции графики и текста.

```
beginfig(1);
% размер рисунка
a=.7in; b=.5in;
% опорные точки
z0=(0,0);
z1=-z3=(a,0);
z2=-z4=(0,b);
% рисование кривой и отрезков
draw z1..z2..z3..z4..cycle;
draw z1--z0--z2;
% метки, привязанные к центрам отрезков
label.top("a", .5[z0,z1]);
label.lft("b", .5[z0,z2]);
dotlabel.bot("(0,0)", z0);
endfig;
```

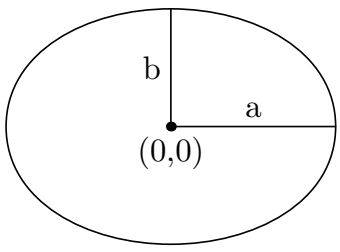

Рис. 2: Овал [1], Figure 18

Для нашего изложения важны строчки кода, содержащие вывод меток - `label` для обозначения полуосей овала. Координаты расположения подписей вычисляются к середины полуосей, а расположение текста в одном случае слева, в другом - сверху от отрезка полуоси. Таким образом, при увеличении масштаба рисунка или изменении соотношения длин осей эллипса, обозначения сохранят свое естественное положение.

Пара `beginfig` (номер иллюстрации в скрипте) и `endfig` обрамляет код `METAPOST` . Далее она не обязательно будет приводиться в коде.

■

Пример 2.2 (Рисование кривых)

По одним и тем же наборам точек можно по-разному провести кривые. В `METAPOST` есть различные способы регулировки кривизны.

```
beginfig(1);
% размер рисунка
a=.7in; b=.5in;
% опорные точки
z0=(0,0);
z1=-z3=(a,0);
z2=-z4=(0,b);
% рисование кривых
draw z1..z2..z3 dashed withdots;
tension_val=2;
draw z1..tension tension_val..z2..tension tension_val..z3;
c=2;
draw z1{curl c}..z2{curl c}..{curl c}z3 dashed evenly;
endfig;
```

Рис. 3: Регулировка кривизны кривой, проведенной через 3 точки

Точками дан случай значений «по умолчанию», что дает окружность. Регулировка с помощью параметра `tension` — сплошная линия, с помощью `curl` — штриховая.

■

Пример 2.3 (Вычисление пересечений прямых [3])

Замечательная возможность METAPOST возможность решения систем уравнений «на лету» для определение геометрических мест. В данном примере вычисляются точки пересечения высот треугольника и противолежащих сторон. Например, в конструкции

```
aa=t1[b,c];  (b-c) dotprod (a-aa)=0;
```

вычисляется точка A_1 пересечения стороны BC нормалью, проведенной из точки A. Неизвестная t_1 параметрически задает положение точки A_1 как доли отрезка BC.

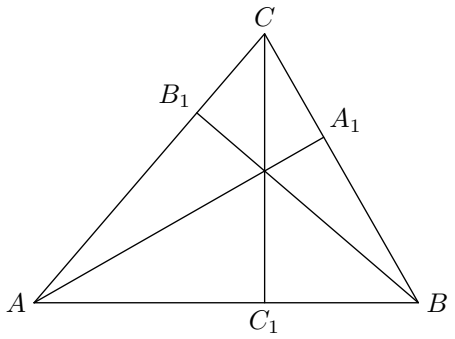

Рис. 4: Высоты треугольника [3], Рис.M.1

```
u:=1 cm;
% объявление переменных - точек на плоскости
pair a, b, c, aa, bb, cc;
numeric t[];
% опороные точки
a:=(0, 0);
b:=(5u,0);
c:=(3u, 3.5u);
% система уравнений для вычисления положения точек A1, B1, C1
aa=t1[b,c];  (b-c) dotprod (a-aa)=0;
bb=t2[a,c];  (a-c) dotprod (b-bb)=0;
cc=t3[a,b];  (a-b) dotprod (c-cc)=0;
% рисование
draw a--b--c--a;
draw a--aa; draw b--bb; draw c--cc;
% метки
label.lft(btex $A$ etex, a);
label.rt(btex $B$ etex, b);
label.top(btex $C$ etex, c);
label.urt(btex $A_1$ etex, aa);
```

```
label.ulft(btex $B_1$ etex, bb);
label.bot(btex $C_1$ etex, cc);
```

В обозначении меток используется LaTeX. Это задается ключевыми словами `btex` и `etex`, между которыми действует синтаксис LaTeX.

■

Пример 2.4 (Построение криволинейной сетки)

Построение сетки. Пусть область ограничена кривой - сплошная линия. Выбираем на ней точки и строим кривые по нормалям в этих точках. Исходную кривую масштабируем с уменьшением.

```
p = z1..z2..z3;
draw p;
lp=length(p);
z5 = point 0.25lp of p;
z6 = point 0.75lp of p;
draw z5{-1,-1}..z6 dashed evenly;
draw p scaled 0.75;
z7 = point 0.125lp of p;
z8 = point 0.875lp of p;
draw z7{-1,-1}..z8 dashed evenly;
```

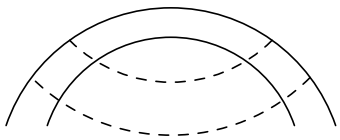

Рис. 5: Построение криволинейной сетки

Получаем разбиение области системой взаимно ортогональных кривых.

■

Пример 2.5 (Выделение замкнутых областей и их закраска, криволинейные стрелки [9])

Выделение замкнутых областей `buildcycle` и их закраска:

```
area1 = buildcycle(p1, E1, p2);
area2 = buildcycle(p3, E1, p4);
fill area1 withcolor red+green;
fill area2 withcolor red+green;
```

криволинейные стрелки - часть объекта `E2`, `subpath`:

```
drawarrow subpath (t3, t4) of E2;
drawarrow subpath (t1, t2) of E2;
```

Сам объект `E2` получен масштабированием овала объекта `E1`.

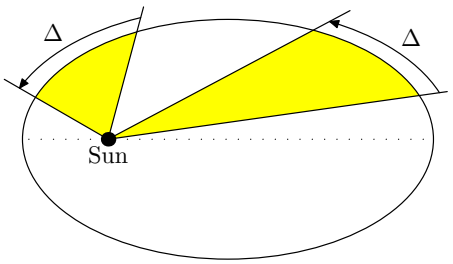

Рис. 6: Keple ellipse [9], exercise 26

Весь пример:

```
u:=1cm; a:=6u; b:=3.5u;
pair sun; sun:=(-1.75u, 0);
path E[], p[], area[];
E1= fullcircle xscaled a yscaled b;
E2 = E1 scaled 1.1;
p1 = sun -- (5u*dir(8) shifted sun);
p2 = sun -- (4u*dir(28) shifted sun);
p3 = sun -- (2u*dir(75) shifted sun);
p4 = sun -- (1.75u*dir(150) shifted sun);
area1 = buildcycle(p1, E1, p2);
area2 = buildcycle(p3, E1, p4);
fill area1 withcolor red+green;
fill area2 withcolor red+green;
draw p1; draw p2; draw p3; draw p4; draw E1;
draw (-a/2,0)--(a/2,0) dashed withdots;
draw sun withpen pencircle scaled 6bp;
label.bot(btex Sun etex, sun);
numeric t[]; % intersection times
t1 = ypart (p1 intersectiontimes E2);
t2 = ypart (p2 intersectiontimes E2);
t3 = ypart (p3 intersectiontimes E2);
t4 = ypart (p4 intersectiontimes E2);
drawarrow subpath (t3, t4) of E2;
drawarrow subpath (t1, t2) of E2;
label.urt(btex $\Delta$ etex, point ((t1+t2)/2) of E2);
label.ulft(btex $\Delta$ etex, point ((t3+t4)/2) of E2);
```

Важно, что криволинейные стрелки суть масштабированный фрагмент выделяемого объекта. То есть этот прием годится для любой формы кривой, часть которой надо выделить на выноске. ■

Пример 2.6 (Метка на криволинейной области [10])

Послев выделения замкнутой области `buildcycle` и создания траектории `pp`, создается объект `label` типа `picture`:

```
picture lab; lab=thelabel(btex $f>0$ etex, z0);
```

В закрашенном объекте командой `unfill` устраняется закраска по размеру `label` и он помещается на освобожденное место.

```
unfill bbox lab; draw lab;
```

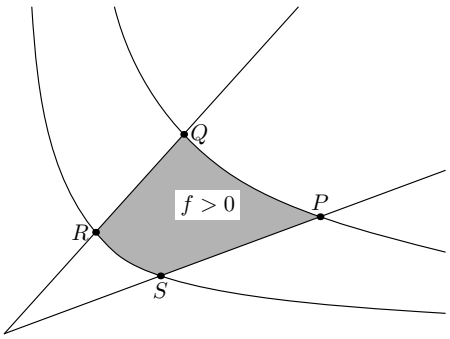

Рис. 7: Пометка области [10]

```
h=2in; w=2.7in;
path p[], q[], pp;
for i=2 upto 4: ii:=i**2;
p[i] = (w/ii,h){1,-ii}...(w/i,h/i)...(w,h/ii){ii,-1};
endfor
q0.5 = (0,0)--(w,0.5h);
q1.5 = (0,0)--(w/1.5,h);
pp = buildcycle(q0.5, p2, q1.5, p4);
fill pp withcolor .7white;
z0=center pp;
picture lab; lab=thelabel(btex $f>0$ etex, z0);
unfill bbox lab; draw lab;
draw q0.5; draw p2; draw q1.5; draw p4;
dotlabel.top(btex $P$ etex, p2 intersectionpoint q0.5);
dotlabel.rt(btex $Q$ etex, p2 intersectionpoint q1.5);
```

```
dotlabel.lft(btex $R$ etex, p4 intersectionpoint q1.5);
dotlabel.bot(btex $S$ etex, p4 intersectionpoint q0.5);
```

Метка криволинейной области в примере расположена по центру прямоугольника, в котором помещается область. ■

Пример 2.7 (Образование множества пересечением нескольких полос. Ограничение области рисования [4])

Пример научной иллюстрации - решение переопределенной системы уравнений с интервально заданными данными. Рисуются оси, 3 полосы, 6 точек их пересечения задают искомое множество, которое закрашивается монохромно.

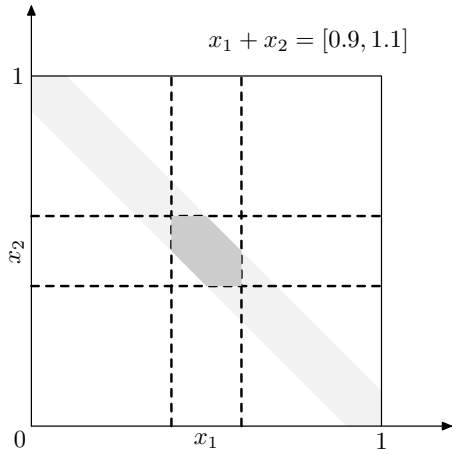

Рис. 8: Пересечение трех полос

```
beginfig(1);
numeric ux, uy; 100ux=10cm; 100uy=10cm;
% переменные
pair A, B, C, D, E, F, Cx, Cxy, Cy, Oxy;
pair k, l, m, n, o, p, q, r, s, t;
numeric Box_size;
% опорные точки и рисование наклонной полосы
Box_size:=50;
A:=(0,45uy); B:=(0, 50uy); C:=(5ux, 50uy);
D:=(50ux, 5uy); E=(50ux, 0); F=(45ux, 0);
fill A--B--C--D--E--F--cycle withcolor .95 white;
% подписи осей и уравнение олной из полос
Cx:=E; Cy:=B; Cxy:=(xpart Cx, ypart Cy); Oxy:=(0ux, 0uy);
label.lft(btex $ 1 $  etex, Cy);
```

```
label.bot(btex $ 1 $  etex, Cx);
label.llft(btex $ 0 $  etex, Oxy);
draw (0,50uy)--(50ux,50uy)--(50ux,0);
drawarrow (0,0)--(0,60uy);
drawarrow (0,0)--(60ux,0);
pickup pencircle scaled 1pt;
label.bot(btex $x_1$  etex, (25ux,0));
label.lft(btex $x_2$  etex rotated 90, (0,25uy));
label.lft(btex $\displaystyle x_1+x_2 = [0.9, 1.1] $ etex,(55ux, 55uy));
% точки 6-угольника и рисование горизонтальный и вертикальной полос
k:= (0.4[Oxy,Cx]--0.4[Cy,Cxy]) intersectionpoint (A--F);
l:= (0.4[Oxy,Cx]--0.4[Cy,Cxy]) intersectionpoint (C--D);
draw (0.4[Oxy,Cx]--0.4[Cy,Cxy]) dashed evenly;
m:= (0.6[Oxy,Cx]--0.6[Cy,Cxy]) intersectionpoint (A--F);
n:= (0.6[Oxy,Cx]--0.6[Cy,Cxy]) intersectionpoint (C--D);
draw (0.6[Oxy,Cx]--0.6[Cy,Cxy]) dashed evenly;
o:= (0.4[Oxy,Cy]--0.4[Cx,Cxy]) intersectionpoint (A--F);
p:= (0.4[Oxy,Cy]--0.4[Cx,Cxy]) intersectionpoint (C--D);
draw (0.4[Oxy,Cy]--0.4[Cx,Cxy]) dashed evenly;
q:= (0.6[Oxy,Cy]--0.6[Cx,Cxy]) intersectionpoint (A--F);
r:= (0.6[Oxy,Cy]--0.6[Cx,Cxy]) intersectionpoint (C--D);
draw (0.6[Oxy,Cy]--0.6[Cx,Cxy]) dashed evenly;
```

Точки пересечения прямых находятся с помощью конструкции `intersectionpoint`.

```
s:= (k--l) intersectionpoint (q--r);
t:= (o--p) intersectionpoint (n--m);
fill o--k--s--r--n--t--cycle withcolor .8 white;
endfig;
```

■

Пример 2.8 (Два рисунка рядом [4])

Продолжение предыдущего примера 2.7, иллюстрирующий технику копирования части иллюстрации для модификации

Фрагмент кода, описывающий перенос ключевых точек левого рисунка вправо с помощью «вектора» `Move` или путем добавки проекции `xpart Move` или оператором `shifted`:

```
pair AA, BB, CC, DD, EE, FF;
pair CCx, CCxy, CCy, OOxy;
pair Move;
% вектор переноса
```

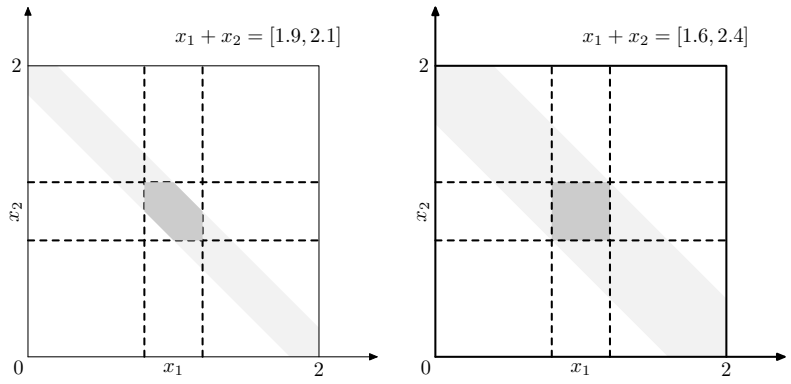

Рис. 9: Два рисунка рядом

```
Move:=(70ux, 0uy);
% перенос на проекцию xpart вектора переноса
AA:=(0+xpart Move,40uy); BB:=(0+xpart Move, 50uy); CC:=(10ux+xpart Move, 50uy);
DD:=(50ux+xpart Move, 10uy); EE=(50ux+xpart Move, 0); FF=(40ux+xpart Move, 0);
fill AA--BB--CC--DD--EE--FF--cycle withcolor .95 white;
```

Альтернативный «векторный» способ переноса лпорных точек и начала координат.

```
CCx:=Cx shifted Move; CCy:=Cy shifted Move;
CCxy:= Cxy shifted Move; OOxy:=Oxy shifted Move;
```

Если бы объект не надо было изменять, его можно было бы скопировать как текущий рисунок, как показано в следующем примере. ■

Пример 2.9 (Использование трансформации. Отражения объекта относительно прямой)

Нужно несколько раз отразить объект от осей.

Строится объект и сохраняется как текущий рисунок `pic:=currentpicture`:

```
pair OO, A, B, C, D;
OO:=(0ux,0uy); A:=(30ux, 40uy); B:=(40ux, 30uy);
draw OO--A--B--cycle;
picture pic;
pic:=currentpicture;
```

Строится отрезок `OC` и относительно него производится отражение текущего рисунка `pic`:

```
% vertical line to reflect
C:=(0ux,60uy);
```

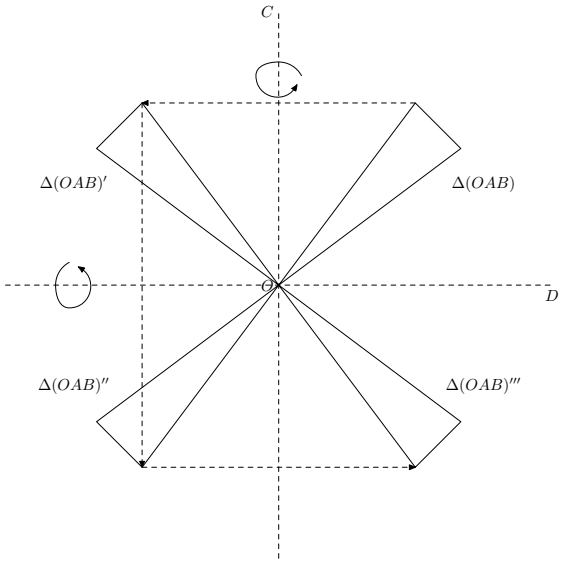

Рис. 10: Отражения треугольника от прямых OC и OD и OC

```
draw (xpart C, -ypart C)--C dashed evenly;
pic:=pic reflectedabout(OO, C);
draw pic;
drawarrow A--(-xpart A, ypart A) dashed evenly;
```

Построение «кривых» стрелок с заданием направления касательных `left, down, ...` в контрольных точках:

```
drawarrow (5ux, 46uy)..(0ux,49uy){left}..(-5ux,46uy){down}..(4ux, 44uy);
drawarrow (-46ux,5uy)..(-49ux,0uy){down}..(-46ux,-5uy){right}..(-44ux, 4uy);
```

■

Пример 2.10 (Касательная к кривой и оператор `whatever` [10])

Кривая c задается двумя точками и касательной в первой точке. Нужно проиллюстрировать построение касательной в точке кривой.

```
path c;
c = (origin){dir 0} .. (150ux,100uy);
draw c;
% выбор точки построения
t:= 0.4;
pair P[];
P1= point t of c;
```

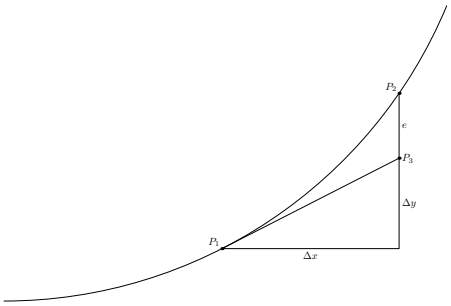

Рис. 11: Касательная к кривой

```
dx:= 60 ux;
(dx, dy) = whatever * direction t of c;
P4 = P1 + (dx,0);
P3 = P4 + (0, dy);
P2 = c intersectionpoint (P4 -- P4+(0, 100uy));
draw P1--P4--P3--cycle;
draw P2--P3;
```

Параметр задает величину касательной к кривой `c`. Точка на кривой с таким значением дается конструкцией

```
P1= point t of c;
```

Для заданного приращения `dx` находится приращение `dy`:

```
dx:= 60 ux;
(dx, dy) = whatever * direction t of c;
```

■

Пример 2.11 (Вращение единичного квадрата с увеличением размера и измененением радиуса окружности)

Опишем вращение единичного квадрата с увеличением размера и измененением радиуса окружности. Исходное положение и параметры изменения размеров и удаления от центра задается следующим образом:

```
scale=10ux;
R:=5scale; % Circle
theta0:=-30; % Rotation step
reduction:=0.9; % Reduction by step
R_step=1.1; % Radius by step
pair start, last_center, center;
```

```
start:=origin shifted (R, 0);
factor:=scale;
center:=start;
```

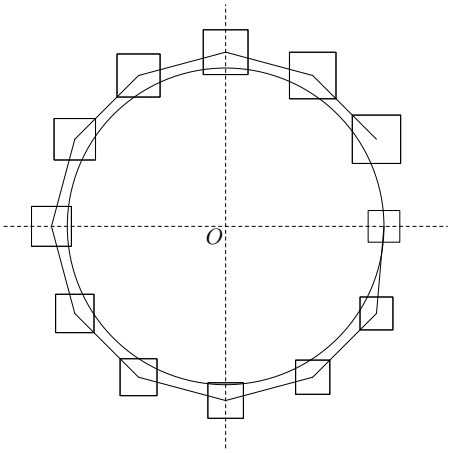

Рис. 12: Вращение квадрата с увеличением размера и удаления от центра вращения

Собственно движение задается циклом:

```
for n=1 upto 11:
last_center:=center;
theta:=-30*n;
factor:=factor*abs(1/cosd(theta0))*reduction;
draw unitsquare scaled  factor  shifted  (-0.5factor, -0.5factor)
shifted (R_step*R*cosd(theta), R_step*R*sind(theta));
center:=origin  shifted (R_step*R*cosd(theta), R_step*R*sind(theta));
draw last_center--center;
endfor;
```

в котором трансформация и положение объекта выписаны в одну строчку:

```
draw unitsquare scaled  factor  shifted  (-0.5factor, -0.5factor)
shifted (R_step*R*cosd(theta), R_step*R*sind(theta));
```

Этот пример в утрированно грубой форме передает картину, тпичную для расчетов с потерей точностью и отклонением от истинного решения, таких как базовая форма метода Эйлера для решения дифференциальных уравнений с начальными условиями.

■

Пример 2.12 (Вращение объекта с увеличением размера - использование цикла)

Представим ситуацию, когда при каждом повороте объекта происходит «переоценка» его размеров. Например, он должен быть вписан в прямоугольник. В таком случае после трансформации его необходимо поместить в «клетку». Для описания минимального размера клетки есть встроенные атрибуты картинки: `llcorner, ulcorner, urcorner lrcorner` и следующая конструкция описывает необходимую «клетку»:

```
uni_square:=llcorner keep--ulcorner keep--urcorner keep--lrcorner keep--cycle;
```

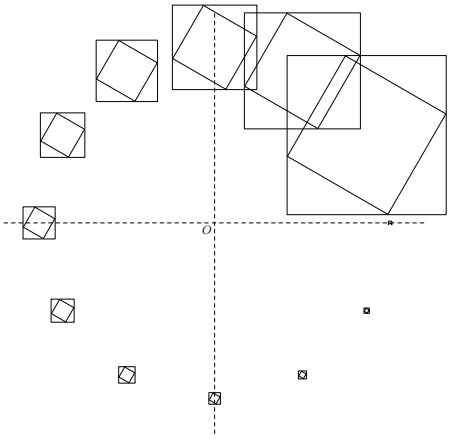

Рис. 13: Вращение квадрата

Полная картина описывается циклом:

```
for i=1 upto 11:
uni_square:=uni_square rotated -30;
draw uni_square;
keep:=currentpicture;
uni_square:=llcorner keep--ulcorner keep--urcorner keep--lrcorner keep--cycle;
draw uni_square;
p[i]:=currentpicture;
currentpicture:=nullpicture;
endfor;
for i=1 upto 11:
draw p[i];
endfor;
```

в котором на каждом шаге текущее изображение записывается в массив и обнуляется:

```
p[i]:=currentpicture;
currentpicture:=nullpicture;
```

■

Пример 2.13 (Использование определений [33])

Найдем пересечение двух объектов примерно круглой формы. Пусть такая форма создается вращением точки по окружности радиуса `r` с отклоненим от идеального положения на случайную величину `(uniformdeviate w, uniformdeviate w)`. Такую функциональность при смещении центра вправо можно описать функцией `vardef randomcirc(expr O, r, w)` следующим образом:

```
vardef randomcirc(expr O, r, w) =
save p,i; pair p; numeric i;
p=O+right*r;
p for i=1 upto 9: .. (p rotatedaround(O, 36i))+(uniformdeviate w, uniformdeviate w)
endfor .. cycle
enddef;
```

Задав аналогично «левую» фигуру `vardef randomcircbis(expr O, r, w)`, можем организовать построение их пересечения pat4 = buildcycle(pat0, pat3):

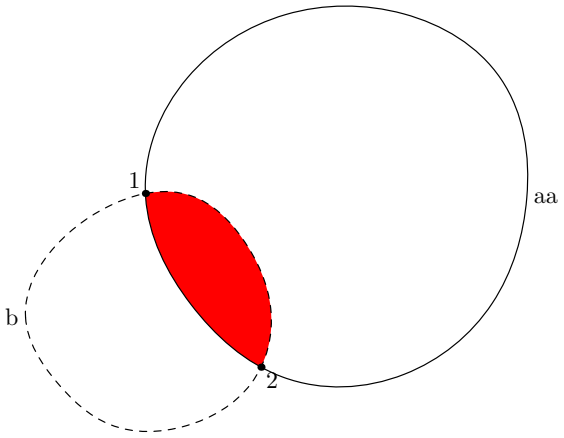

Рис. 14: Использование определений

```
u:=10pt;
vardef randomcirc(expr O, r, w) =
save p,i; pair p; numeric i;
p=O+right*r;
p for i=1 upto 9: .. (p rotatedaround(O, 36i))+(uniformdeviate w, uniformdeviate w)
endfor .. cycle
enddef;
%
```

```
vardef randomcirc_bis(expr O, r, w) =
save p,i; pair p; numeric i;
p=O+left*r;
p for i=1 upto 9: .. (p rotatedaround(O, 36i))+(uniformdeviate w, uniformdeviate w)
endfor .. cycle
enddef;
%
path pat[];
pair p[];
p0:=(0,5u); p3:=(-8u,0);
pat0:=randomcirc(p0,8u,.5u);
pat3:=randomcirc_bis(p3,5u,0.4u);
pat4 = buildcycle(pat0, pat3);
fill pat4 withcolor red;
draw pat0; draw pat3 dashed evenly;
```

■

Пример 2.14 (Пересечение траекторий, использование `intersectiontimes` [34])

В `METAPOST` при нахождении геометрического места пересечения двух параметризованных объектов (прямая, кривая) можно определить параметры с помощью команды `intersectiontimes`.

Нйдем одно из пересечений диагоналей прямоугольника и круга, находящегося на пересечении диагогалей. Создадим дополнительный треугольник, одна из вершин которого находится в центре круга.

```
path rectangle, triangle, circle;
numeric A, B, M;
A=89; B=144; M=21;

rectangle = unitsquare yscaled A xscaled B;
triangle  = subpath (0,1) of rectangle --
center rectangle -- cycle;
circle = fullcircle scaled M shifted center rectangle;

draw rectangle;
draw triangle;
draw circle;
```

Решение с помощью `intersectionpoint`

```
draw subpath (0,6) of circle withcolor red;
pair p;
p = triangle intersectionpoint subpath (4,6) of circle;
dotlabel.lft(btex $p$ etex, p);
```

Решение с помощью `intersectiontimes`

```
numeric t, u;
pair pt;
(t,u) = circle intersectiontimes triangle;
```

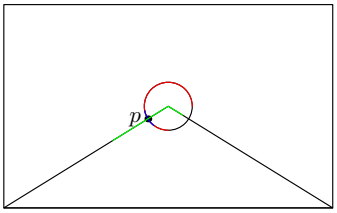

Рис. 15: Пересечение траекторий

В выражении

```
(t,u) = circle intersectiontimes triangle;
```

величины `(t,u)` - суть параметры круга и треугольника, соответствующие решению. На рисунке 15 синим и зеленым цветом показаны части круга и треугольника, соотвествующие диапазонам изменения параметров кривых `(t,u)` в пределах 10% изменений этих параметров.

```
draw subpath (0.9t,1.1t) of circle withcolor blue;
draw subpath (0.9u,1.1u) of triangle withcolor green;
```

■

Существует еще множество мощных и элегантных приемов работы с METAPOST . Они достаточно хорошо документированы [9], [10], [8]. Репозиторий METAPOST для LaTeX- ресурс CTAN [2].

Пример 2.15 (Рисование поверхности [9])

Несмотря на то, что METAPOST ориентирован на рисование 2D-иллюстраций, можно перейти и в третье измерение. Рассмотрим пример для поверхности

$$z(x, y) = cos(x \cdot y)$$

Объявление переменных, базового цвета, точки наблюдения (xp, yp, zp) и системы отсчета (Xr,Yr,Zr) :

```
% u: dimensional unit
% xp, yp, zp: coordinates of light source
% bf : brightness factor
% base_color : base color
numeric u,xp,yp,zp,bf;
color base_color;
u = 1cm;
xp := 3; yp := 3; zp := 5;
bf := 30;
base_color := red+green;
% O, Xr, Yr, Zr : reference frame
pair O,Xr,Yr,Zr;
O = (0,0);
Xr = (-.7,-.7) scaled u;
Yr = (1,0) scaled u;
Zr = (0,1) scaled u;
```

Рисование системы координат:

```
vardef frameXYZ(expr s) =
drawarrow O--Xr scaled s;
drawarrow O--Yr scaled s;
drawarrow O--Zr scaled s;
label.llft(btex $X$ etex scaled 1.25, (Xr scaled s));
label.rt(btex $Y$ etex scaled 1.25, (Yr scaled s));
label.top(btex $Z$ etex scaled 1.25, (Zr scaled s));
enddef;
```

От 3D-координат к 2D. Вычисление частных производных по x и y.

```
vardef project(expr x,y,z) = x*Xr + y*Yr + z*Zr enddef;
% numerical derivatives by central differences
vardef diffx(suffix f)(expr x,y) =
numeric h; h := 0.01;
(f(x+h,y)-f(x-h,y))/(2*h)
enddef;
vardef diffy(suffix f)(expr x,y) =
numeric h; h := 0.01;
(f(x,y+h)-f(x,y-h))/(2*h)
enddef;
```

Вычисление яркости в узлах сетки от источника света:

```
vardef brightnessfactor(suffix f)(expr x,y,z) =
numeric dfx,dfy,ca,cb,cc;
dfx := diffx(f,x,y);
```

```
dfy := diffy(f,x,y);
ca := (zp-z)-dfy*(yp-y)-dfx*(xp-x);
cb := sqrt(1+dfx*dfx+dfy*dfy);
cc := sqrt((z-zp)*(z-zp)+(y-yp)*(y-yp)+(x-xp)*(x-xp));
bf*ca/(cb*cc*cc*cc)
enddef;
```

Вычисление цветов и рисование «заплаток»:

```
vardef z_surface(suffix f)(expr xmin,xmax,ymin,ymax,nx,ny) =
numeric dx,dy,xt,yt,zt,factor[][];
pair Z[][];
dx := (xmax-xmin)/nx;
dy := (ymax-ymin)/ny;
for i=0 upto nx:
xt := xmin+i*dx;
for j=0 upto ny:
yt := ymin+j*dy;
zt := f(xt,yt);
Z[i][j] = project(xt,yt,zt);
factor[i][j] := brightnessfactor(f,xt,yt,zt);
endfor
endfor
for i = 0 upto nx-1:
for j= 0 upto ny-1:
fill Z[i][j]--Z[i][j+1]--Z[i+1][j+1]--Z[i+1][j]--cycle
withcolor factor[i][j]*base_color;
endfor
endfor
enddef;
```

Объявление целевой функции, области определения переменных, числа узлов сетки, положения источника света, яркости и масштаба - рис. 16:

```
beginfig(1);
xp := 3;
yp := 3;
zp := 10;
bf := 100;
numeric pi; pi := 3.14159;
vardef cos primary x = cosd(x/pi*180) enddef;
vardef f(expr x,y) = cos(x*y) enddef;
z_surface(f,-3,3,-3,3,100,100);
frameXYZ(5);
label(btex $z = \cos(xy)$ etex scaled 1.25,(0,-4cm));
endfig;
```

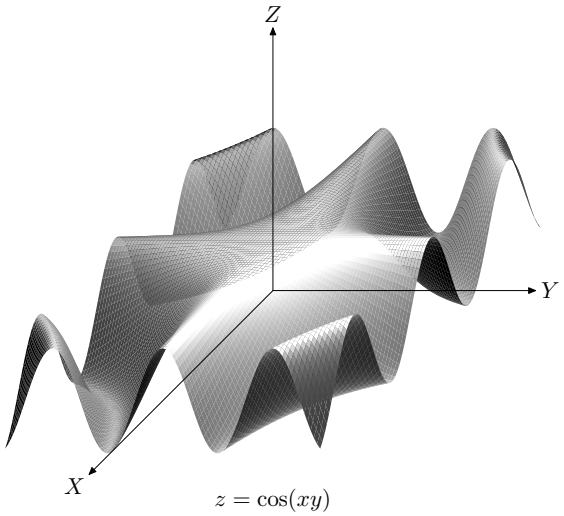

Рис. 16: Функция двух переменных

В итоге, получилась необходимая поверхность. Успех достигнут, но код, конечно, большой. Особенно это важно в тех случаях, когда нужно подобрать расположение источника освещения и угол зрения. ■

3 Featpost - 3D-Расширение Metapost

Для большинства иллюстраций в пособиях автора достаточно 2D-иллюстраций. Для трехмерных изображений существует 3D-расширение `METAPOST` - FEATPOST [16]. Это очень мощное средство, к сожалению, не имеет удобной справочной системы.

Подключение возможностей FEATPOST производится добавлением в преамбулу документа вызова:

```
input featpost3Dplus2D;
```

Поскольку исходная философия `METAPOST` двумерна, необходим объект, имеющий три компоненты для описания пространственных величин. В FEATPOST используется встроенный тип color:

```
(red,green, blue)
```

При этом значения компонент могут иметь произвольные значения.

Пример 3.16 (Прямые, стрелки, конус в Featpost [17].)
Рассмотрим построение конуса и аннотаций:

Рисование прямой и стрелок 3D

```
draw rp(color1)--rp(color2);
drawarrow rp(color1)--rp(color2);
```

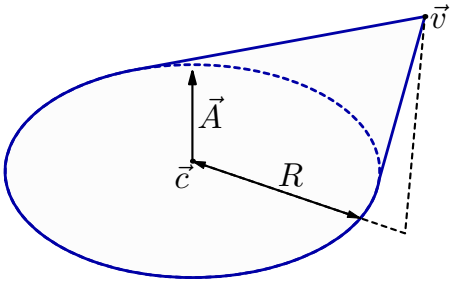

Рис. 17: Конус

Рисование конуса [16]

```
verygoodcone(ShadowOn,basecenter,basenormal,reflen,vertex);
```

Параметры `basecenter,basenormal,vertex` имеют тип color.
Параметр `ShadowOn` булев, управляет видимостью ■

Пример 3.17 (3 вида перспективы - кубики [26].)

Функция `makeface` создает грань из точек `V`. Функция `makeline` создает линию из точек `V`.

```
f := ( 1.2 , 2.0 , 1.6 );
Spread := 75;
V1 := (1,1,1);
V2 := (-1,1,1);
V3 := (-1,-1,1);
V4 := (1,-1,1);
V5 := (1,1,-1);
V6 := (-1,1,-1);
V7 := (-1,-1,-1);
V8 := (1,-1,-1);
makeface1(1,2,3,4);makeface2(5,6,7,8);
makeface3(1,2,6,5);makeface4(2,3,7,6);
makeface5(3,4,8,7);makeface6(4,1,5,8);
makeline1(1,7);makeline2(2,8);
makeline3(3,5);makeline4(4,6);
```

Центральная проекция:

```
beginfig(2);
ParallelProj := false;
SphericalDistortion := false;
draw_all_test(true);
```

```
endfig;
```

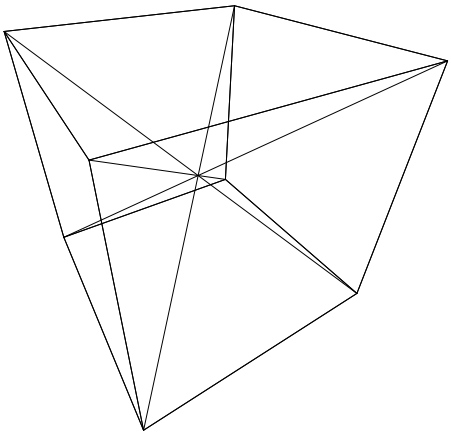

Рис. 18: cube central central

Параллельная проекция:

```
beginfig(1);
ParallelProj := true;
SphericalDistortion := false;
pickup pencircle scaled 1pt;
draw_all_test(true);
endfig;
```

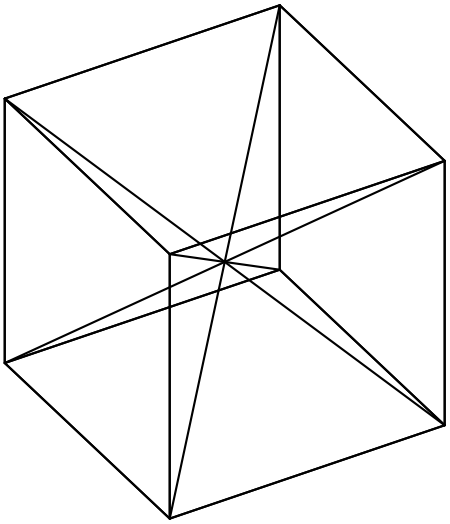

Рис. 19: cube perspective parallel

Сферическая проекция:

```
beginfig(3);
ParallelProj := false;
SphericalDistortion := true;
PrintStep := 5;
draw_all_test(true);
endfig;
```

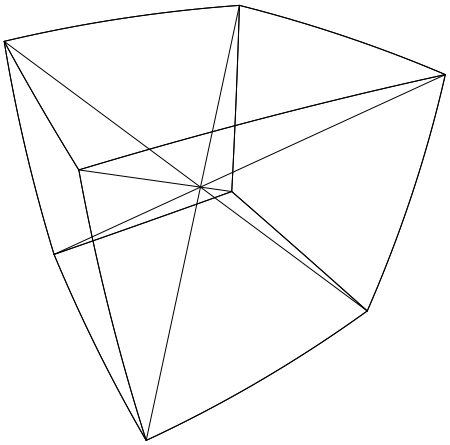

Рис. 20: cube spherical parallel

■

Пример 3.18 (3 вида перспективы - проекции [25].)

Общее построение. Точка фокуса f, переменные, размеры четырехугольника:

```
f := (4,2,2.5);
Spread := 45;
color psdf, psdu, psdv, rll, rlr, rur, rul, pll, plr, pur, pul;
color pc;
numeric projecplanesize, sizefraction, positfraction, aspectratio, rectsize;
pen realpen, lightpen, projpen, dotpen;
path projecplanepath, realpath;
psdf = 3.1*(0,1,0.37);
projecplanesize = 1.7;
sizefraction = 0.19;
positfraction = 0.65;
aspectratio = 1.0;
rectsize = 0.15;
realpen = pencircle scaled 2.6pt;
lightpen = pencircle scaled 0.55pt;
```

```
projpen = pencircle scaled 1.8pt;
dotpen = pencircle scaled 3.8pt;
```

Плоскости и проекция:

```
psdv = N( (-Y(psdf), X(psdf), 0) );
psdu = ncrossprod( psdf, psdv );
pll = projecplanesize*(-aspectratio*psdv-psdu);
plr = projecplanesize*(+aspectratio*psdv-psdu);
pur = projecplanesize*(+aspectratio*psdv+psdu);
pul = projecplanesize*(-aspectratio*psdv+psdu);
projecplanepath = rp(pll)--rp(plr)--rp(pur)--rp(pul)--cycle;
pc = positfraction*psdf;
rll = pc+sizefraction*projecplanesize*(-psdv-psdu);
rlr = pc+sizefraction*projecplanesize*(+psdv-psdu);
rur = pc+sizefraction*projecplanesize*(+psdv+psdu);
rul = pc+sizefraction*projecplanesize*(-psdv+psdu);
realpath = rp(rll)--rp(rlr)--rp(rur)--rp(rul)--cycle;
```

Параллельная проекция:

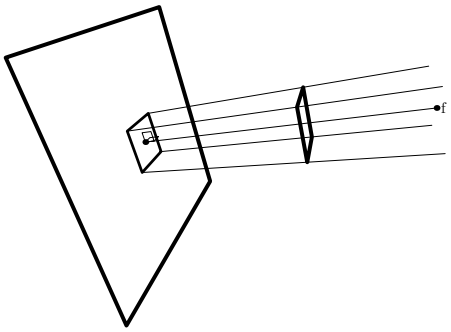

Рис. 21: perspective parallel

```
beginfig(1);
draw projecplanepath withpen realpen;
draw realpath withpen realpen;
draw rp(psdf) withpen lightpen;
drawoptions( withpen lightpen );
squareangline( psdf, psdv, black, rectsize );
squareangline( psdf, psdu, black, rectsize );
drawoptions();
pair prll, prlr, prur, prul;
prll = rp(rll-pc);
prlr = rp(rlr-pc);
prur = rp(rur-pc);
```

```
prul = rp(rul-pc);
path projpath;
projpath = prll--prlr--prur--prul--cycle;
draw projpath withpen projpen;
draw prll--rp(rll+(1-positfraction)*psdf) withpen lightpen;
draw prlr--rp(rlr+(1-positfraction)*psdf) withpen lightpen;
draw prur--rp(rur+(1-positfraction)*psdf) withpen lightpen;
draw prul--rp(rul+(1-positfraction)*psdf) withpen lightpen;
draw rp(black)--rp(psdf) withpen lightpen;
draw rp(black) withpen dotpen;
draw rp(psdf) withpen dotpen;
label.rt( btex f etex, rp(psdf) );
%  label.bot( btex viewcentr etex, rp(black) );
endfig;
```

Центральная проекция:

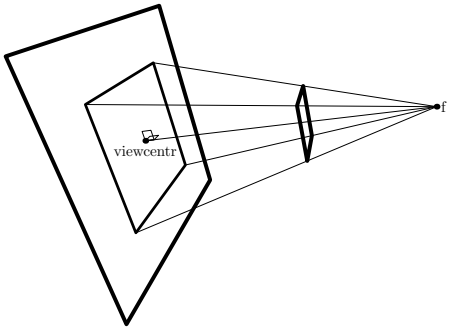

Рис. 22: perspective central

Сферическая проекция: ■

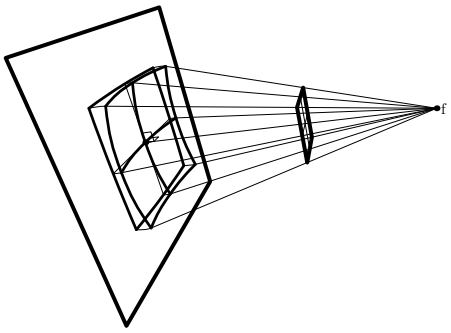

Рис. 23: perspective spherical

4 Элементы Metapost в Python PyX

METAPOST вдохновляет разработчиков разрабатывать похожие средства. Например, язык векторной графики Asymptote [11]. Это более современный язык векторной графики, ориентированный на использование с C++ и Python и расширенный на 3D-графику. По существу, Asymptote - полная альтернатива METAPOST . В связи с этим, рассмотрим более частные возможности.

Пример 4.19 (Кривые в стиле METAPOST в PyX [13])

Библиотка PyX позволяет строить различные примитивы векторной графики. Пример на кривые Безье, открытые (openpath) и замкнутые (closedpath).

```
from pyx import *
from pyx.metapost.path import beginknot, endknot, smoothknot, tensioncurve

p1, p2, p3, p4, p5 = (0, 0), (2, 1.33), (1.3, 3), (0.33, 2.33), (1, 1.67)
openpath = metapost.path.path([
beginknot(*p1), tensioncurve(), smoothknot(*p2), tensioncurve(),
smoothknot(*p3), tensioncurve(), smoothknot(*p4), tensioncurve(),
endknot(*p5)])
closedpath = metapost.path.path([
smoothknot(*p1), tensioncurve(), smoothknot(*p2), tensioncurve(),
smoothknot(*p3), tensioncurve(), smoothknot(*p4), tensioncurve(),
smoothknot(*p5), tensioncurve()])
c = canvas.canvas()
for p in [p1, p2, p3, p4, p5]:
c.fill(path.circle(p[0], p[1], 0.05), [color.rgb.red])
c.fill(path.circle(p[0], p[1], 0.05), [color.rgb.red, trafo.translate(2, 0)])
c.stroke(openpath)
c.stroke(closedpath, [trafo.translate(2, 0)])
```

После построения, процесс завершается сохранением результата в один из форматов: EPS, PDF или SVG.

```
c.writeEPSfile("metapost")
c.writePDFfile("metapost")
c.writeSVGfile("metapost")
```

■

Пример 4.20 (Пружины и параллельные кривые в PyX [14])

Одной из возможностей PyX является модификация имеющихся примитивов. Такие преобразования называются деформациями

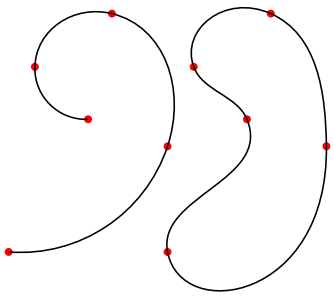

Рис. 24: Кривые в стиле METAPOST в PyX

```
from pyx import *
n = 3                  # number of masses
r = 3.0                # system radius
rcyc = 0.3             # radius of cycloid
nl = 13                # number of loops
rc = 0.5               # radius of masses
eps = 0.03             # extra spacing for surrounding circles

c = canvas.canvas()
springcircle = path.circle(0, 0, r)
masspositions = [i*springcircle.arclen()/n
for i in range(n)]
for springsegment in springcircle.split(masspositions):
c.stroke(springsegment,
[deformer.cycloid(rcyc, nl),
deformer.smoothed(radius=0.1)])
for x, y in springcircle.at(masspositions):
c.fill(path.circle(x, y, rc))
```

В частности, в данном строятся две кривые, параллельные окружности, заданной выражением `springcircle = path.circle(0, 0, r)`.

```
c.stroke(springcircle, [deformer.parallel(rc+eps)])
c.stroke(springcircle, [deformer.parallel(-rc-eps)])
```

Собственно базовая окружность `springcircle` не показана на рисунке, она проходит через центры масс. Отображены «деформированные» окружности, с отступом наружу и внутрь на `rc+eps`.

■

Пример 4.21 (Узел в PyX [15])

Пример узла с ипользованием параллельных кривых. Базовая кривая.

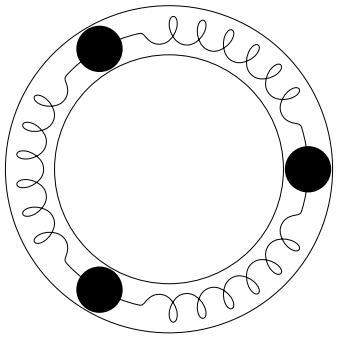

Рис. 25: Пружины и параллельные кривые PyX

```
from pyx import *
unit.set(uscale=3, wscale=3)
dist = 0.15           # расстояние между канатами
thick = 0.08          # толщина канатов
# Построение базовой линии каната. Это кривая лежащая между ними.
A = 1.0, 0.1    # точка, где заканчивается прямая часть каната и начинается узел
B = 1.3, 0      # точка, где узел загибается, накрывая прямую часть
t = -0.8, -0.1  # направление в точке A
s =  0.3, 1.2   # направление в точке B
# точки
pts = [
( A[0]-t[0], A[1]-t[1]), ( A[0]     , A[1]     ),
( A[0]+t[0], A[1]+t[1]), (-B[0]-s[0],-B[1]-s[1]),
(-B[0]     ,-B[1]     ), (-B[0]+s[0],-B[1]+s[1]),
( B[0]-s[0], B[1]-s[1]), ( B[0]     , B[1]     ),
( B[0]+s[0], B[1]+s[1]), (-A[0]-t[0],-A[1]-t[1]),
(-A[0]     ,-A[1]     ), (-A[0]+t[0],-A[1]+t[1]) ]
# Базовая кривая
seam = path.path(
path.moveto(*(pts[0])),
path.lineto(*(pts[1])),
path.curveto(pts[2][0],pts[2][1],pts[3][0],pts[3][1],pts[4][0],pts[4][1]),
path.curveto(pts[5][0],pts[5][1],pts[6][0],pts[6][1],pts[7][0],pts[7][1]),
path.curveto(pts[8][0],pts[8][1],pts[9][0],pts[9][1],pts[10][0],pts[10][1]),
path.lineto(*(pts[11])) )
# Небольшое сглаживание для лучшего согласования curveto и lineto
seam = deformer.smoothed(0.6).deform(seam)
```

Нахлест канатов

```
# The ropes, when drawn later, will have to overlap in a very specific way.
l = seam.arclen()
```

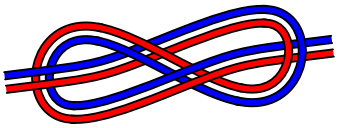

Рис. 26: Узел в PyX

```
epsilon = 0.002*l # малое перекрытие сегментов
seam_segs = seam.split([0.28*l+epsilon, 0.28*l-epsilon,
0.42*l+epsilon, 0.42*l-epsilon,
0.58*l+epsilon, 0.58*l-epsilon,
0.72*l+epsilon, 0.72*l-epsilon])[::2]
# Для каждого сегмента, две смещенных траетории строят границы каждой веревки
ropes_segs = []
for seam_seg in seam_segs:
ropes_segs.append([])
for ropeshift in [-0.5*dist, 0.5*dist]:
ropes_segs[-1].append([])
for edgeshift in [-0.5*thick, 0.5*thick]:
ropes_segs[-1][-1].append(
deformer.parallel(ropeshift + edgeshift).deform(seam_seg))
# ropes_segs - это список сегментов, содержащий список канатов, каждый из которых содержит
rope_colors = [color.rgb.blue, color.rgb.red]
c = canvas.canvas()
# Поскольку сегменты должны перекрываться вполне определенным образом
# мы должны нарисовать их в правильном порядке
for index in [1, 4, 2, 0, 3]:
for rope_seg, col in zip(ropes_segs[index], rope_colors):
seg = rope_seg[0].joined(rope_seg[1].reversed())
seg.append(path.closepath())
c.fill(seg, [col])
c.stroke(rope_seg[0], [style.linecap.round, style.linejoin.round])
c.stroke(rope_seg[1], [style.linecap.round, style.linejoin.round])
```

■

5 Научные и технические приложения METAPOST

В заключение рассмотрим некоторые применения METAPOST . Существует большое количество надстроек и библиотек для рисования огромного количества типов деловых, научных и технических иллюстраций. Выбор примеров продиктован исключительно вкусом автора.

5.1 Электрические схемы

Библиотека mpcirc позволяет рисовать электрические схемы, как аналоговые, так и цифровые.

Пример 5.22 (Источник AC и LC-контур в mpcirc [35])

Источник AC и LC-контур.

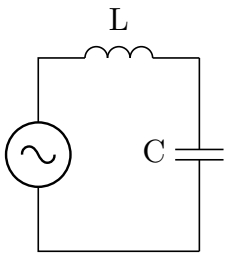

Рис. 27: Источник AC и LC-контур

```
u:=10bp; % unit of length
input mpcirc;
beginfig(1);
prepare(L,C,Vac); % mention your elements
z0=(10u,10u); % lower right node
ht:=6u; % height of circuit
z1=z0+(0,ht); % upper right node
C=.5[z0,z1]; % location of capacitor
L.t=T.r; % use default orientation
C.t=Vac.t=T.u; % components rotated 90 degrees
% set the distance between Voltage and Capacitor
equally_spaced(5u,0) Vac, C;
L=z1-0.5(C-Vac); % location of spool
edraw; % draw components of the circuit
% draw wires connecting components
% the first ones rotated 90 degrees
wire.v(Vac.a,z0);
wire.v(Vac.b,L.a);
```

```
wire.v(L.b,z1);
wire(C.a,z0);
wire(C.b,z1);
```

■

Пример 5.23 (Источник AC и последовательный RLC-контур в mpcirc [35])

Источник AC и последовательный RLC-контур.

```
u:=10bp; % unit of length
input mpcirc;
beginfig(1);
prepare (L,R,C,Vac); % mention your elements
z0=(0,0); % lower left node
ht:=6u; % height of circuit
z1=z0+(0,ht); % upper left node
Vac=.5[z0,z1]; % location of voltage
Vac.t=T.u; % rotated 90 degrees
L.t=R.t=C.t=T.r; % default orientation
% set equal distances
equally_spaced(5u,0) z1,R,L,C,z2;
edraw; % draw elements of circuits
% draw wires connecting nodes
% the first ones rotated 90 degrees
wire.v(Vac.a,z0);
wire.v(Vac.b,z1);
wire.v(z2,z0);
wire(z1,R.a);
wire(R.b,L.a);
wire(L.b,C.a);
wire(C.b,z2);
endfig;
```

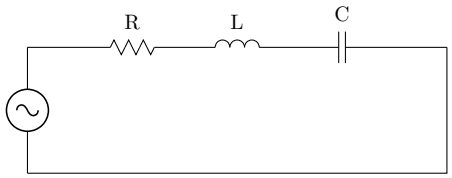

Рис. 28: Источник AC и последовательный RLC-контур

■

Пример 5.24 (Источник AC и смешанный RLC-контур в mpcirc [35])

Источник AC и смешанный RLC-контур.

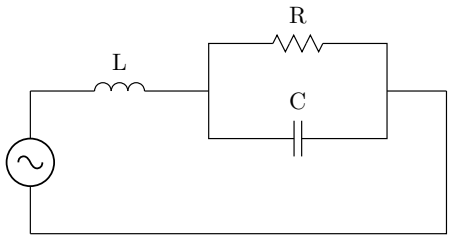

Рис. 29: Источник AC и смешанный RLC-контур

```
u:=10bp; % unit of length
input mpcirc;
prepare (L,R,C,Vac); % mention your elements
z0=(0,0); % lower left node
ht:=6u; % height of circuit
z1=z0+(0,ht); % upper left node
Vac=.5[z0,z1]; % location of voltage
Vac.t=T.u; % rotated 90 degrees
L.t=R.t=C.t=T.r; % default orientation
% set equal distances
equally_spaced(7.5u,0) z1,z2,z3;
L=0.5[z1,z2]; % location of spool
C=0.5[z2,z3]-(0,2u);
R=0.5[z2,z3]+(0,2u);
z4 = z3+(2.5u,0);
edraw; % draw elements of circuits
% draw wires connecting nodes
wire.v(Vac.a,z0);
wire.v(Vac.b,z1);
wire(z1,L.a);
wire(L.b,z2);
wire.v(z2,C.a);
wire.v(z2,R.a);
wire.v(z3,C.b);
wire.v(z3,R.b);
wire(z3,z4);
wire.v(z4,z0);
```

■

Мощный пакет - circuitikz [36], часть Tikz [39] см также [37].

5.2 Физика

Физические иллюстрации.

Пример 5.25 (Векторы в 3D [31])

Пример рисования векторов в 3D с надписями

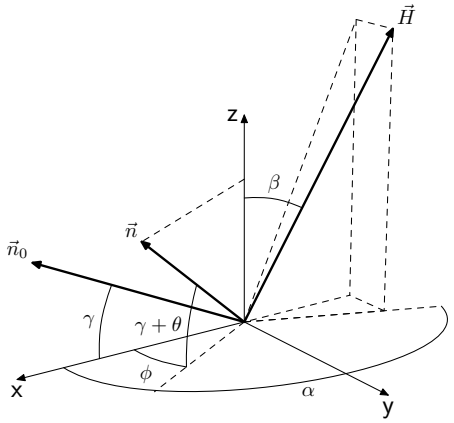

Рис. 30: Векторы в 3D

```
    f := 3*(1.2,1.7,0.8);

Spread := 125;

color origr, nzero, nothr, ndxyl, vmagf, mfxzp, mfxyp, mfxxp;
numeric alphan, gamman, thetan, phiang, mfsize, axxc, ayyc, azzc;
numeric phiarmlen, alphafac;
axxc = 0.99;
ayyc = 0.87;      % make this 1.28 when in twist-bend geometry
azzc = 0.82;
alphan = 170;     % make this 90 to get the twist-bend geometry
gamman = 25;
thetan = 10;
phiang = 30;
mfsize = 1.25;
phiarmlen = 0.75;
alphafac = 1.4;
origr = (0,0,0);
```

```
nzero = (cosd(gamman),0,sind(gamman));
ndxyl = phiarmlen*(cosd(phiang),sind(phiang),0);
nothr = (cosd(phiang)*cosd(gamman+thetan),
sind(phiang)*cosd(gamman+thetan),sind(gamman+thetan));
vmagf = mfsize*(cosd(alphan)*sind(gamman),sind(alphan),
-cosd(alphan)*cosd(gamman));
mfxzp = (X(vmagf),0,Z(vmagf));
mfxyp = (X(vmagf),Y(vmagf),0);
mfxxp = (X(vmagf),0,0);
cartaxes( axxc, ayyc, azzc );
draw rp(origr)--rp(ndxyl) dashed evenly;
draw rp(origr)--rp(mfxzp)--rp(vmagf) dashed evenly;
draw rp(origr)--rp(mfxyp)--rp(vmagf) dashed evenly;
draw rp(origr)--rp(alphafac*mfxyp) dashed evenly;
draw rp(mfxzp)--rp(mfxxp)--rp(origr) dashed evenly;
draw rp(mfxyp)--rp(mfxxp) dashed evenly;
draw rp((0,0,Z(nothr)))--rp(nothr) dashed evenly;
angline(nzero,(1,0,0),origr,0.65,btex $\gamma$ etex,lft);
angline(ndxyl,(1,0,0),origr,0.5,btex $\phi$ etex,llft);
angline(ndxyl,nothr,origr,0.5,btex $\gamma+\theta$ etex,lft);
angline(vmagf,(0,0,1),origr,0.5,btex $\beta$ etex,top);     % twist-bend
angline(mfxyp,(1,0,0),origr,0.8,btex $\alpha$ etex,bot);
pickup pencircle scaled 1.15pt;
drawarrow rp(origr)..rp(nzero);
drawarrow rp(origr)..rp(nothr);
drawarrow rp(origr)..rp(vmagf);
label.ulft(btex $\vec{n}_0$ etex,rp(nzero));
label.ulft(btex $\vec{n}$ etex,rp(nothr));
label.urt(btex $\vec{H}$ etex,rp(vmagf));
endfig;
```

■

Пример 5.26 (Каустики [27])

Каустика (от греческого *καυστικοζ*, жгучий) — огибающая семейства лучей, не сходящихся в одной точке. Каустики в оптике — это особые линии (в двухмерном случае) и особые поверхности, вблизи которых резко возрастает интенсивность светового поля.

Отражение света от поверхности.
s - луч, a - угол падения, p -поверхность, l - расстояние

```
vardef reflectrayr(expr s,a,p,l)=
save tI,tn,ia,I,J;
```

```
pair I,J;
tI=xpart(p intersectiontimes ((s+1mm*dir(a))--(s+30cm*dir(a))));
if tI>0:
I=point tI of p;
draw s--I;
tn=angle(direction tI of p)+90;
ia:=tn-angle(s-I);
reflectrayr(I,tn+ia,p,l);
else:
J=s+dir(a)*l;
drawarrow s--J;
fi;
enddef;
```

Сферическая поверхность.

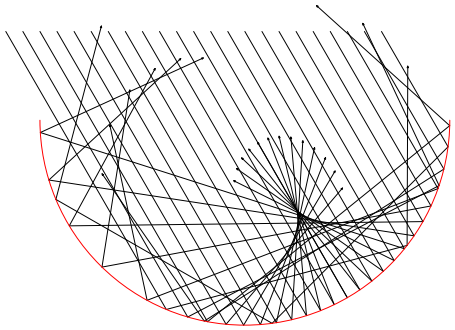

Рис. 31: Каустика от сферической поверхности

```
beginfig(3); % spheric dish
numeric u, i, num, incidang;
pair sunpoi;
u = 8cm;
num = 12;
incidang = -60;
path p;
p = halfcircle scaled 4u rotated 180 shifted (0,u);
for i=1 upto 2num-1:
sunpoi := ((-2+2*i/num,1)*u)-u*dir(incidang);
reflectrayr(sunpoi,incidang,p,14cm);
endfor;
draw p withcolor red;
endfig;
```

■

Пример 5.27 (Работа линзы [28])

Иллюстрация работы линзы. Ключевой вопрос - описание преломления лучей на границе двух сред.

Цвета для различных лучей

```
numeric specialdist, otherdist;
specialdist = 2.5cm;
color TableC[];
TableC0 := (0.65,0.61,0.49);    % grey        %% G N U P L O T
TableC1 := 0.77red;             % red         %%
TableC2 := ( 0.2, 0.2, 1.0 );   % blue        %%    colors
TableC3 := ( 1.0, 0.7, 0.0 );   % orange      %%
TableC4 := 0.85green;           % pale green  %%
TableC5 := 0.90*(red+blue);     % magenta     %%
TableC6 := 0.85*(green+blue);   % cyan        %%
TableC7 := 0.85*(red+green);    % yellow      %%
```

Преломление света в линзе - `lensrefractray(point,angle,lenseleftsurface,lenserightsurface,rir,color)`.

Входные параметры:
point, angle - источник света
lenseleftsurface, lenserightsurface - левая (плоская) и правая (выпуклая) поверхности линзы,
rir - относительный показатель преломления

Выходные параметры:
I - точка пересечения луча с линзой слева
J - конечная точка трассировки луча
K - точка пересечения луча с линзой справа
sib - внутренний угол выхода луча из линзы
ib - угол выхода луча из линзы
ib - угол луча внутри линзы после плоской поверхности
tI - параметр точки I плоской поверхности
tn - нормаль в точке I

```
vardef lensrefractray(expr s,a,p,q,rir,c)=
save tI,tn,ia,ib,I,J,K,sib;
pair I,J,K;
draw s--(s-2cm*dir(a)) withcolor TableC[c];
tI = xpart(p intersectiontimes (s--(s+15cm*dir(a))));
if tI>=0:
I=point tI of p;
draw s--I withcolor TableC[c];
```

```
tn=angle(direction tI of p)+90;
ia=tn-angle(s-I);
sib=sind(ia)/rir;
ib=-angle(1+-+sib,sib);
J=I+dir(tn+180+ib)*2cm;
tI := xpart(q intersectiontimes (I--J));
if tI>=0:
K:=point tI of q;
draw I--K withcolor TableC0;
tn:=angle(direction tI of q)+90;
ia:=tn-angle(I-K);
if sind(ia)<1/rir:
sib:=rir*sind(ia);
ib:=angle(1+-+sib,sib);
J:=K-dir(tn-ib)*specialdist;
draw K--J withcolor TableC[c];
fi;
fi;
fi;
enddef;
```

Построение линзы

```
numeric rir, ray, wid, dist, ha, dx, dy, bord, cx, cy, cent, a, sa, nr, m;
numeric refang, colorcounter;
path vseg, cseg, lens, cell;
pen fordot;
color lenscolor, cellcolor;
refang = 41;
m = 1.2mm;
nr = 9;
rir = 1.2;
ray = 2.3cm;
wid = 3.9cm;
dist = 1.1cm;
bord = 1mm;
fordot = pencircle scaled 1mm;
lenscolor = 0.6white;
cellcolor = 0.4white;
dy = 0.5*wid;
dx = ray+-+dy;
cx = 0.5*bord;
cy = dy-bord;
cent = dx-dist;
```

```
ha = angle(dist,cy);
sa = 2ha/nr;
z1 = (dx,ray);
z2 = (dx,-ray);
z3 = (cent,0);
z4 = (dx,cy);
z5 = (dx,-cy);
z6 = (0,cy);
z7 = (0,-cy);
z8 = (ray,cy);
z9 = (ray,-cy);
vseg = z2--z1;
%cseg = halfcircle rotated -90 scaled 2ray;
cseg = (dx,-dy){dir(refang)}..(ray,0)..{dir(180-refang)}(dx,dy);
lens = buildcycle(reverse vseg,z7--z9,cseg,z8--z6);
fill lens withcolor lenscolor;
cell = ((cx,cy)--(-cx,cy)--(-cx,-cy)--(cx,-cy)--cycle) shifted z3;
fill cell withcolor cellcolor;
```

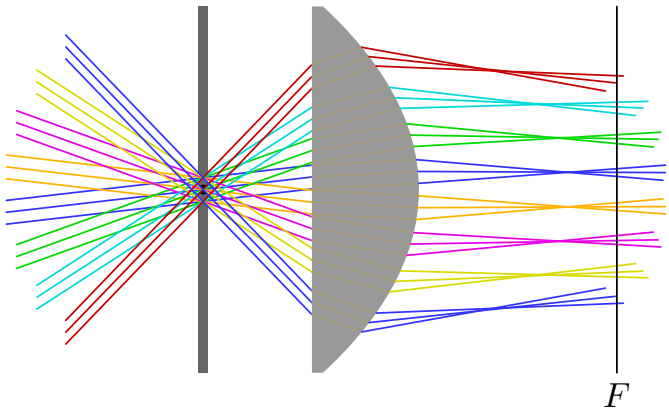

Рис. 32: Работа линзы - перефокусировка. Показатель преломления 2.2

Трассировка

```
colorcounter = 1;
for a=0.5*sa step sa until ha-sa:
if not (colorcounter<>7):
colorcounter:= 1;
else:
colorcounter := incr(colorcounter);
fi;
lensrefractray(z3,a,vseg,cseg,1.9,colorcounter);
lensrefractray(z3+up*m,a,vseg,cseg,1.9,colorcounter);
```

```
lensrefractray(z3+down*m,a,vseg,cseg,1.9,colorcounter);
if not (colorcounter<>7):
colorcounter:= 0;
else:
colorcounter := incr(colorcounter);
fi;
lensrefractray(z3,-a,vseg,cseg,1.9,colorcounter);
lensrefractray(z3+up*m,-a,vseg,cseg,1.9,colorcounter);
lensrefractray(z3+down*m,-a,vseg,cseg,1.9,colorcounter);
endfor;
otherdist = ray+specialdist-0.5cm;
clip currentpicture to (-otherdist,-cy)--(-otherdist,cy)--(otherdist,cy)--
(otherdist,-cy)--cycle;
```

Меняя показатель преломления, можно подобрать такое значение, при котором будет достигнута фокусировка в плоскости.

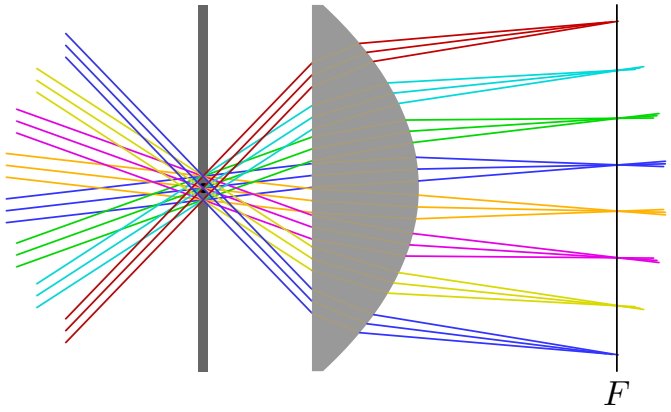

Рис. 33: Работа линзы - точная фокусировка. Показатель преломления 1.9

Подобным образом работает хрусталлик человеческого глаза [29]. ■

Пример 5.28 (Преломление света через две поверхности [30].)

Иллюстрация преломления света через две поверхности. Первая поверхность задается как кривая, проходящая через 4 точки. Вторая - как отражение и сдвиг первой.

```
numeric u;
u=3mm;
path p;
p=((5u,-6u)..(4u,-2u)..(4.5u,0)..(5u,5u)) rotated 90;
path q;
```

```
q = reverse (p yscaled (-1)) shifted (up*10.5u);
fill (p--q--cycle) withcolor red;
```

Трассировка

```
for a=46 step 3 until 120:
lensrefractray(origin,a,p,q,1.4);
endfor;
endfig;
```

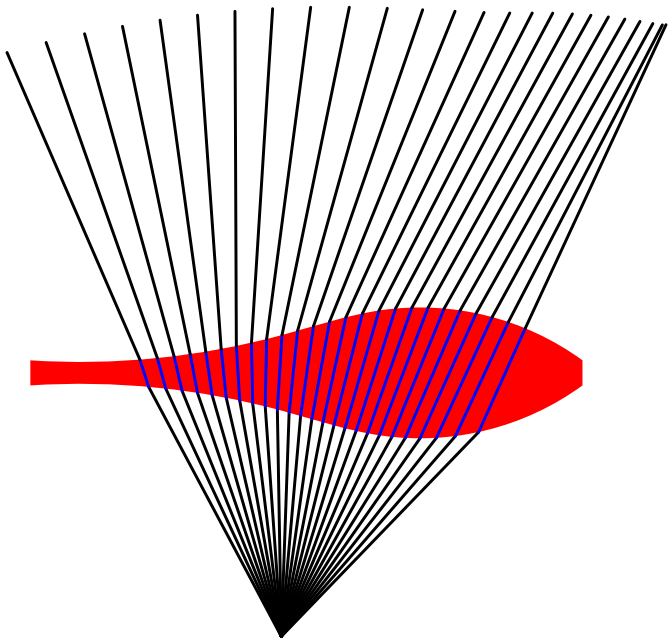

Рис. 34: Иллюстрация преломления света через две поверхности. Показатель преломления 1.2

Левая часть «линзы» практически плоская и лучи в этой части мало отклоняются. В правой части происходит фокусировка. ■

Пример 5.29 (Камера-обскура [20].)

Пример достаточно сложной конструкции. Камера-обскура (лат. camera obscūra — «тёмная комната») — простейший вид устройства, позволяющего получать оптическое изображение объектов. Представляет собой светонепроницаемый ящик с отверстием в одной из стенок и экраном (матовым стеклом или тонкой белой бумагой) на противоположной стене. Лучи света, проходя сквозь малое отверстие (диаметр (зависит от «фокусного расстояния» камеры, приблизительно 0,1—5 мм) создают перевёрнутое изображение на экране.

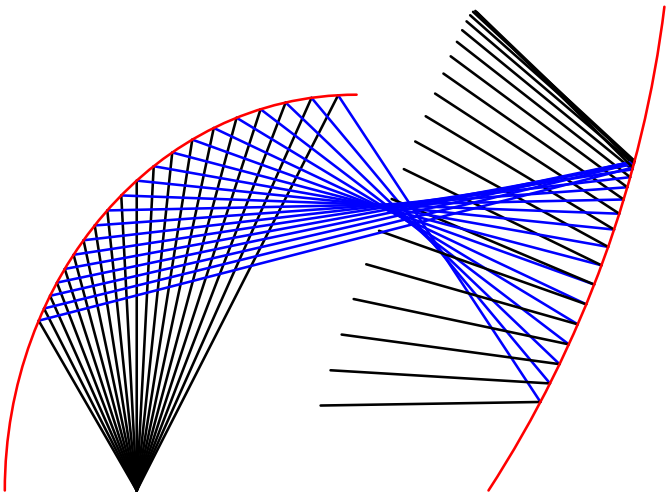

Рис. 35: Иллюстрация отражения света от двух поверхностей.

Также используется при регистрации рентгеновских лучей, например при регистрации излучения плазмы.

Задание геометрии наблюдения

```
input featpost3Dplus2D;
beginfig(1);
f := (10,7,8);
Spread := 45;
```

Задание размеров конструкции.
Камера - w(idth), h(eight), l(ength).
Диафрагма - d(iameter), t(ube length), we - положение.
Отступы для стрелок - tr, tl, ma, mb.
Угол зрения диафрагмы - ar.

```
numeric w, h, l, d, t, tr, tl, ma, mb, we, ar;
w = 3;h = 2;l = 4;d = 0.75;t = 1.2;tr= 0.15;tl= 0.25;ma= 0.3;mb= 0.2;
ar= 2;we= 0.6*(h-t);
pen thin, thick, aver;
thin = pencircle scaled 0.5pt;
thick= pencircle scaled 1.5pt;
aver = pencircle scaled 1.0pt;
ahlength:=2mm;
ahangle:=30;
color p[];

p1 = (0,w/2,-ma-mb);
p2 = (0,w/2,-ma);
```

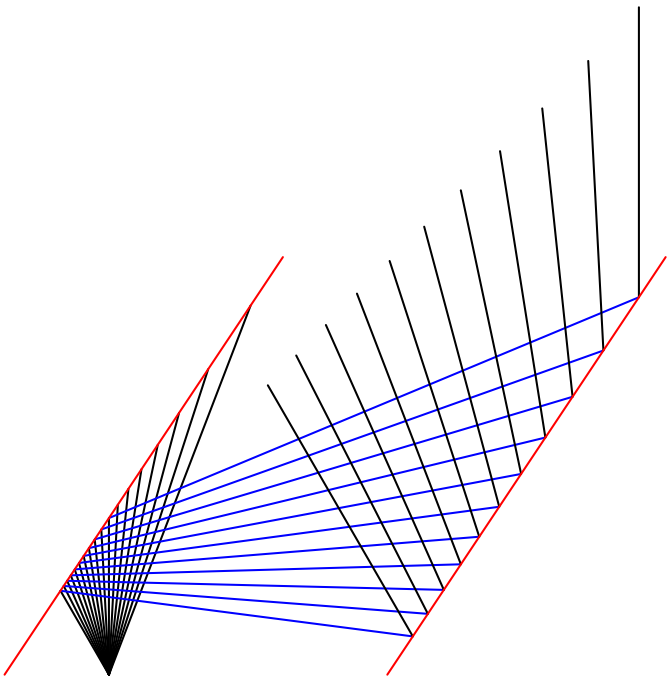

Рис. 36: Иллюстрация отражения света от двух зеркал.

```
p3 = (0,w/2,0);
p4 = (0,w/2,h);
p5 = (0,-w/2,-ma-mb);
p6 = (0,-w/2,-ma);
p7 = (0,-w/2,0);
p8 = (0,-w/2,h);
p9 = (0,-w/2-ma-mb,0);
p10= (0,-w/2-ma,0);
p11= (0,-w/2-ma-mb,h);
p12= (0,-w/2-ma,h);
p13= (-l,-w/2-ma-mb,h);
p14= (-l,-w/2-ma,h);
p15= (-l,-w/2,h);
p16= (-l,-w/2,0);
p17= (-l,w/2,0);
p18= (-l,w/2,h);
p19= (-l+d,0,0);
p20= (-l+d,0,t);
p21= (-l-ma,0,t);
p22= (-l-ma-mb,0,t);
p23= (-l-ma,0,0);
```

```
p24= (-l-ma-mb,0,0);
p25= (-l+d,w/2+ma,0);
p26= (-l+d,w/2+ma+mb,0);
p27= (-l,w/2+ma,0);
p28= (-l,w/2+ma+mb,0);
p29= 0.5[p2,p6];
p30= 0.5[p10,p12];
p31= 0.5[p12,p14];
p32= 0.5[p21,p23];
p33= 0.5[p25,p27];
p34= (0,0,h);
p35= (0,0,2*t-h);
```

Собственно рисование: отрезки

```
drawoptions( withpen thick );
draw rp(p8)--rp(p4)--rp(p3)--rp(p7)--rp(p8)--rp(p15)--rp(p16);
draw rp(p15)--rp(p18)--rp(p17)--rp(p3)--rp(p4)--rp(p18);
path pat[];
pat1 = rp(p8)--rp(p4);
pat2 = rp(p7)--rp(p16)--rp(p17);
pat3 = rp(p4)--rp(p18);
pat4 = pat2 cutbefore pat1 cutafter pat3;
draw pat4;
drawoptions( withpen thin );
```

Стрелки для размеров и каркас

```
drawdblarrow rp(p2)--rp(p6);
drawdblarrow rp(p10)--rp(p12);
drawdblarrow rp(p14)--rp(p12);
drawdblarrow rp(p21)--rp(p23);
drawdblarrow rp(p25)--rp(p27);
draw rp(p1)--rp(p3);
draw rp(p5)--rp(p7);
draw rp(p7)--rp(p9);
draw rp(p8)--rp(p11);
draw rp(p13)--rp(p15);
draw rp(p20)--rp(p22);
draw rp(p19)--rp(p24);
draw rp(p17)--rp(p28);
```

Точка наблюдения f=(10,7,8) - рисунок 37

Рисование диска, эллипса, дуги обозначения угла и меток

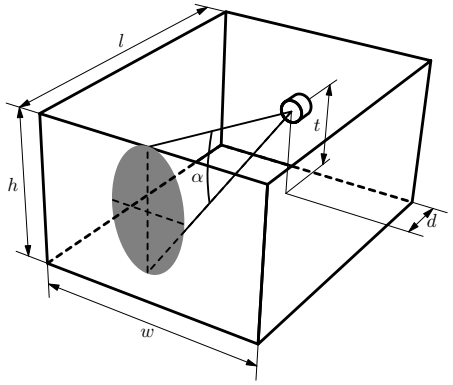

Рис. 37: Рентгеновская камера. Точка наблюдения f=(10,7,8)

```
draw rp(p19)--rp(p26);
drawoptions( withpen thick );
rigorousdisc( 0, true, p20, tr, (-tl,0,0) );
draw rp(p19)--rp(p20) withpen thin;
drawoptions( withpen aver );
draw rp(p34)--rp(p20)--rp(p35);
fill ellipticpath( (0,0,t), (h-t)*blue, we*green ) withcolor 0.5white;
draw pat2 dashed evenly withpen thick;
draw rp(p20)--rp(p35)--rp(p34) dashed evenly;
draw rp((0,0,t)-we*green)--rp((0,0,t)+we*green) dashed evenly;
angline( p34, p35, p20, ar, btex $\alpha$ etex, llft );
label.bot(btex $w$ etex, rp(p29));
label.lft(btex $h$ etex, rp(p30));
label.top(btex $l$ etex, rp(p31));
label.lft(btex $t$ etex, rp(p32));
label.rt(btex $d$ etex, rp(p33));
endfig;
```

Изменяем точку наблюдения - «на уровень пола». Пусть f=(10,7,0) - рисунок 38

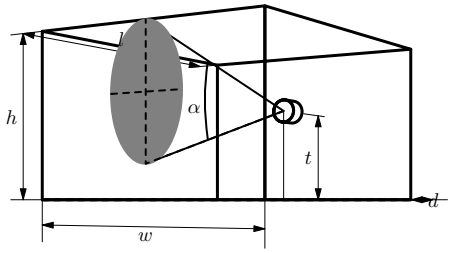

Рис. 38: Рентгеновская камера. Точка наблюдения f=(10,7,0)

Задание разных точек наблюдения позволяет строить «обход» объекта. ■

Пример 5.30 (Гравитационное рассеяние антиматерии или опыт Резерфорда [24].)

Шуточный пример про рассеяние антигравитирующих частиц и одновременно демонстрация вполне реального опыта по рассеянию заряженных частиц на ядрах атомов - классический опыт Резефорда [23].

Задание переменных и констант:

```
f := (30,0,0);
Spread := 100;

numeric initdist, timestep, Gfactor, vprobe;
inidist = 10; timestep = 0.1;
Gfactor = 2;
vprobe = 2.2;
numeric numsteps, numtries, diststep, i, sigma, sigmang, angl;
color inipos, inivel, dstp, veloc;
path traject;
pen bluepen, redpen;
bluepen = pencircle scaled 0.9mm;
redpen = pencircle scaled 0.55mm;
```

Задание закона отталкивания от рассеивающего центра в векторной форме:

```
def gravitfield( expr relatposprobe ) =
( Gfactor*relatposprobe/((conorm( relatposprobe ))**3) )
enddef;
```

Как показал еще Ньютон, в поле силы, спадающей по закону обратных квадратов, тела движутся по коническим сечениям. В случае неограниченного движения это гиперболы и парабола.

Задание движения по параболе:

```
vardef parabol( expr index ) =
traject:=trajectorypath(
numsteps, inipos+index*dstp, inivel, timestep)(gravitfield);
draw traject;
enddef;
```

Задание движения по гиперболе:

```
vardef hiperbo( expr veloc ) =
traject:=trajectorypath(
numsteps, inipos, veloc, timestep)(gravitfield);
draw traject;
enddef;
```

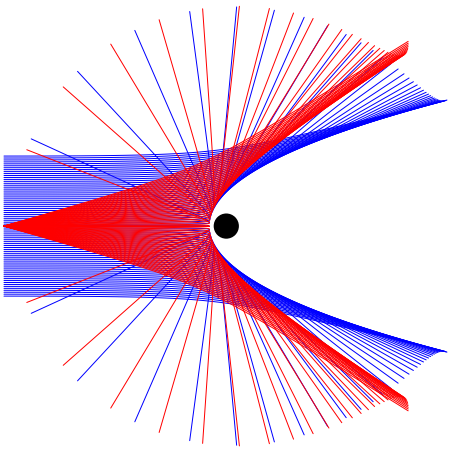

Рис. 39: Движение тела в поле отталкивающего потенциала

Собственно рисование - перебор начальных условий

```
beginfig(1);
numtries = 36;
numsteps = 95;
sigma = 3.15;
sigmang = 16;
diststep = sigma/numtries;
dstp = (0,0,diststep);
inipos = (0,-inidist,0);
inivel = (0,vprobe,0);
drawoptions( withpen bluepen withcolor blue );
for i=-numtries upto numtries:
parabol( i );
endfor;
drawoptions( withpen redpen withcolor red );
for i=-numtries upto numtries:
angl := i*sigmang/numtries;
veloc := vprobe*(0,cosd(angl), sind(angl));
hiperbo( veloc );
endfor;
drawoptions();
fill fullcircle scaled (4cm) shifted (rp(black));
produce_auto_scale;
endfig;
```

■

Пример 5.31 (Движение частиц в торе [31].)

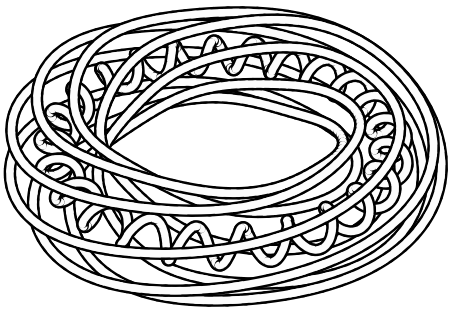

Рис. 40: Траектории в торе

Рисование различных видов траекторий в торе.

```
input featpost3Dplus2D;

% Draw a torus with NP dots, a section of 3.14*RS^2
% and a total diameter of 2*(RB+RS). The dots are placed
% on a single closed line that turns around the $z$ axis
% NB times and around the section NS times.
% figure(1) exemplifies the use of signalvertex.

def flask( expr TheVal ) =
begingroup
numeric RB, RS, RP, NB, NS, NM, phi, theta, first, second, third;
RB=1;
RS=0.3;
RP=0.02;
NB=8;
NS=5;
NM=4;
theta=360*NB*TheVal;
phi=360*NS*TheVal;
first =(RB+(RS+RP*sind(NM*theta))*cosd(phi))*cosd(theta);
second=(RB+(RS+RP*sind(NM*theta))*cosd(phi))*sind(theta);
third =    (RS+RP*sind(NM*theta))*sind(phi);
( (first,second,third) )
endgroup
enddef;

def blask( expr TheVal ) =
```

```
begingroup
numeric RB, RS, NB, NS, phi, theta, first, second, third;
RB=1;
RS=0.1;
NB=1;
NS=24;
theta=360*NB*TheVal;
phi=360*NS*TheVal;
first =(RB+RS*cosd(phi))*cosd(theta);
second=(RB+RS*cosd(phi))*sind(theta);
third =    RS*sind(phi);
( (first,second,third) )
endgroup
enddef;

beginfig(5);
Spread := 200;
numeric NP, i, varfrac;
path cl;
string comm;
pen bip, sp;
varfrac = 1.25;
bip = pencircle scaled 15pt;
sp = pencircle scaled 9pt;
NP=600;
cl = for i=1 upto NP: rp(flask(i/NP)).. endfor cycle;
draw cl;
for i=1 upto NP:
comm:="draw subpath ("
& decimal(i-0.5)
& ","
& decimal(i+0.5)
& ") of cl withpen bip; undraw subpath ("
& decimal(i-varfrac)
& ","
& decimal(i+varfrac)
& ") of cl withpen sp;";
getready( comm, flask( i/NP ) );
endfor;
closedline( true, 300, 0.5, 1 )( blask );
doitnow;
endfig;
```

■

Пример 5.32 (Правило правой руки [42].)

Немного про векторное произведение «на пальцах».

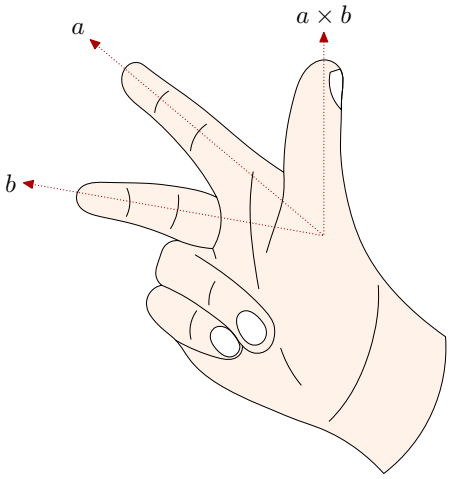

Рис. 41: Правило правой руки

Задание параметров кисти

```
vardef left_hand(expr shade) =
save H, N;
path H, N[];

H = (28,-10)..(37.5,0)..(52,9.6)..(60,13.1)..(70,18)..(74,20)..(91,30)
..(101,40)..(106,48)..(110,52)..(115,58)..(116,60)..(116.4,64)..(110,70)
& (110,70)..(112,74)..(110,81)..(108,85)..(102,87)
& (102,87)..(110,91)..(120,95)..(130,98)..(136,100)..(140,103)..(140.4,109)
..(130,112)..(121,111.2)..(110,110)..(100,108)..(90,105)..(86,103)
& (86,103)..(90,111)..(95,119)..(103.5,130)..(112,139)..(120,150)..(113,158)..
(110,156)..(91,140)..(70,120)..(59,111)
& (59,111)..(58,120)..(56,130)..(53,140)..(48,150)..(40,154)..(34,150)..(34,140)..
(35,130)..(36,120)..(36,110)..(35,100)..(30,80)..(10,51)..(0,42)
&(0,42) {down} .. (28,-10) & cycle;

N0 = buildcycle(H, ((34.5,133.8) .. (38,140) .. (38,149) & (38,149) .. (34,150)));
N3 = (74,44)..(81,50)..(80,57)..(71,52){dir 240}..cycle;
N4 = (85,40)..(90,43)..(92,50)..(90,52)..(84,50)..(81,44)&(81,44)..cycle;
```

```
save p; picture p; p = image(
fill H withcolor .9[shade, white];

unfill N0; draw N0;
unfill N3; draw N3;
unfill N4; draw N4;

draw (59,111) { down }..(60,101)..(64,90)..(68,80);

draw (102,146) .. (108,138) {dir -80};
draw (88,132) .. (95,122) { dir -80};
draw (71,112) .. (73,100) .. (72,66) { dir -96};

draw (121,109)..(122,98) { dir -59 };
draw (101,105)..(105,92) { dir -54 };
draw (86,103).. { dir -54 } (89,83) & (89,83) .. (102,87) {2,1};
draw (89,83)..{dir -100}(88,79);

draw (110,70)..(100,64)..(90,56)..(84,50)..(80,44)..(76,41)..
(70,42)..(67,48)..(67,51)..(70,57) ..(80,68)..(96,81);
draw (89,70)..(92,61){dir -90};

draw (106,48)..(90,40)..(82,40)..(80,44);
draw (97,59)..(100,48){dir -100};
draw (65,42)..(60,30)..(58,27);

draw H;

draw (25,64){down}.. (30,39) .. (48,12){dir -40};
) shifted -(45,85) rotated -4; p
enddef;

vardef right_hand(expr shade) =
left_hand(shade) reflectedabout(up,down)
enddef;
```

Рисование векторов

```
beginfig(1);

draw right_hand(red + 1/2 green + 1/8 blue);

path a[];
a1 = origin -- 80 up;
```

```
a2 = origin -- 120 up rotated 50;
a3 = origin -- 120 up rotated 80;

for i=1 upto 3:
drawarrow a[i] dashed withdots scaled 1/4 withcolor 2/3 red;
endfor

label.top (btex $a \times b$ etex, point 1 of a1);
label.ulft(btex $a$ etex, point 1 of a2);
label.lft (btex $b$ etex, point 1 of a3);

endfig;
end.
```

■

5.3 Геометрия

Пример 5.33 (Теорема Морлея [43].)

Собственно, все примеры так или иначе связаны с геометрией. Однако, пусть будет и чисто геометрический пример! Одна из последних теорем планиметрии - XX век! Имеет применение в трилинейных координатах
Теорема.
«Точки пересечения смежных трисектрис углов произвольного треугольника являются вершинами правильного (равностороннего) треугольника».

Построение трисектрис.

```
vardef premiere_trisectrice (expr M, A,B,C) =
(M-A) = whatever * ( (A-B) rotated 1/3 (angle(C-A) - angle(B-A)) );
draw A--M;
enddef;
vardef deuxieme_trisectrice (expr M, A,B,C) =
(M-A) = whatever * ( (A-B) rotated 2/3 (angle(C-A) - angle(B-A)) );
draw A--M;
enddef;
```

Определение точек треугольника $\Delta(M_1, M_2, M_3)$:

```
pair A,B,C,M[];
u:=2cm;
A=(0,0); B=(2u,-.5u); C=(u,u);
draw A--B--C--cycle;
```

```
premiere_trisectrice(M1,A,B,C);
deuxieme_trisectrice(M1,B,C,A);
premiere_trisectrice(M2,B,C,A);
deuxieme_trisectrice(M2,C,A,B);
premiere_trisectrice(M3,C,A,B);
deuxieme_trisectrice(M3,A,B,C);
```

Рисование

```
draw M1--A; draw M1--B;
draw M2--B; draw M2--C;
draw M3--C; draw M3--A;
draw M1--M2--M3--cycle;
```

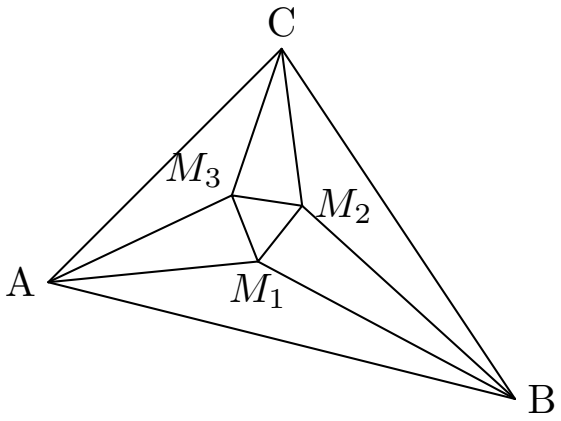

Рис. 42: Теорема Морлея

Треугольник $\Delta(M_1, M_2, M_3)$ - равносторонний.

■

5.4 Механика

Пример 5.34 (Сверло Рело[44].)

«Круглый треугольник» Рело (F. Reuleaux) - кривая постоянной ширины [45].

Кривой постоянной ширины называется плоская выпуклая кривая, расстояние между любыми двумя параллельными опорными прямыми которой постоянно. Стороны квадрата — опорные прямые: каждая сторона касается треугольника, но не пересекает его. Треугольник Рёло можно вращать, и при этом он всегда будет касаться каждой стороны квадрата; таким образом ширина треугольника (расстояние между двумя опорными прямыми) постоянна.

Имеет много применений в технике — на его основе были созданы кулачковые и грейферные механизмы, роторно-поршневой двигатель Ванкеля и инструменты, позволяющие сверлить (фрезеровать) квадратные отверстия.

Построение

```
  numeric u, desv, ray;
path rouletri, pathpart, pa, pb, pc, border;
u = 150mm;
desv = u*sqrt(3)/3;
% величина смещение центра вращения
ray = (sqrt(3)/3-0.5)*u/2;
pathpart = halfcircle scaled u cutafter (origin--u*dir(60));
pathpart := pathpart shifted (desv*dir(-150)/2);
pa = pathpart;
pb = pathpart rotated 120;
pc = pathpart rotated -120;
rouletri = pa--pb--pc--cycle;
draw rouletri shifted (ray*down) withcolor redwithpen pencircle scaled 2bp;
border=(u/4,u/4)--(-u/4,u/4)--(-u/4,-u/4)--(u/4,-u/4)--cycle;
```

«Сверление квадрата»:

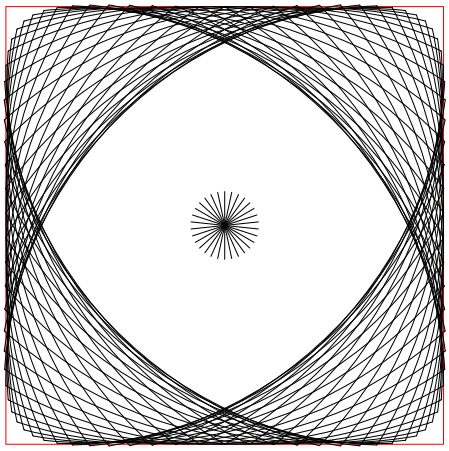

Рис. 43: Сверло Рело

```
border=(u/4,u/4)--(-u/4,u/4)--(-u/4,-u/4)--(u/4,-u/4)--cycle;
draw border shifted (ray*up);
draw border withcolor red;
numeric i, N, astep;
N = 4;
astep = 3*N;
for i=astep step astep until 360:
```

```
% смещение центра вращения ray*dir(i-90)
draw rouletri rotated (-i/3) shifted (ray*dir(i-90));
endfor;
```

Концы отрезков из центра квадрата - положения мгновенного центра крепления «треугольника». ■

Пример 5.35 (Двигатель Ванкеля [47].)

Продолжение темы фигур постоянной ширины. Двигатель Ванкеля [48] - роторно-поршневой двигатель — роторный двигатель внутреннего сгорания. Особенность двигателя — применение трёхгранного ротора (поршня), имеющего вид треугольника Рело, вращающегося внутри цилиндра специального профиля, поверхность которого выполнена по эпитрохоиде - плоской кривой, образуемой точкой, жёстко связанной с окружностью, катящейся по внешней стороне другой окружности.

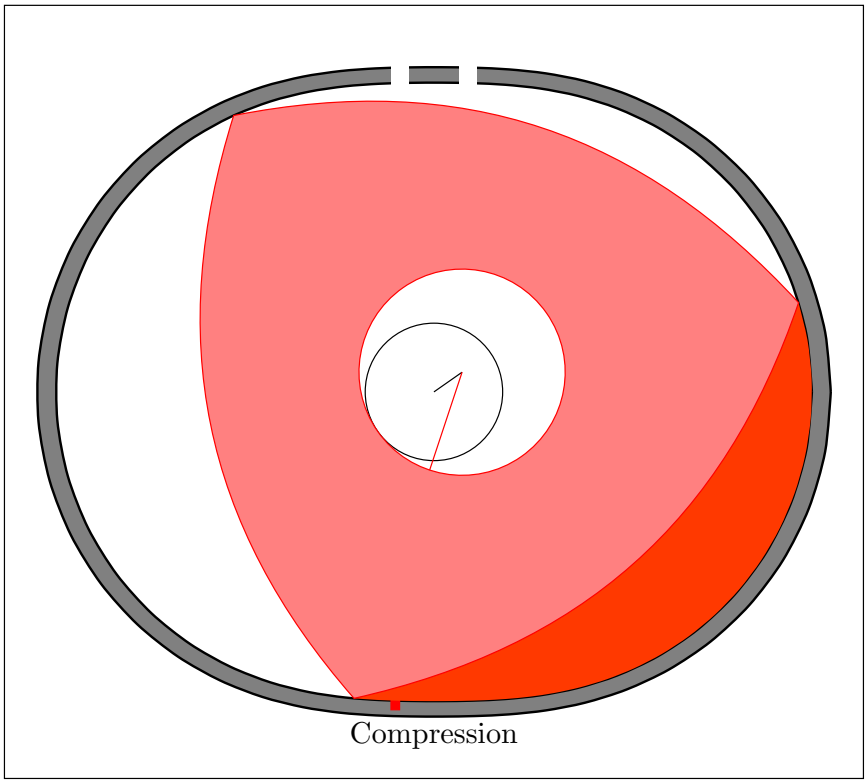

Рис. 44: Двигатель Ванкеля

Преамбула - подключение библиотек calc, animate, fp.

```
\usetikzlibrary{calc}
\usepackage{animate}
\usepackage{fp}
% Useful for calculating the position of the different phases
of the motor but useless for simulation only
```

Двигатель Ванкеля - ротор

```
\newcommand{\Wankel}[1]{
% \def\itheta{#1}
\FPabs{\itheta}{#1}
% dAⓒfinition des donnAⓒes \ data definition
\def\OA{0.4}
\def\OB{0.8}
\def\AE{4}
% \def\seuil{500}
% \def\couleur{20}
```

Углы характерных точек - мест впуска нового топлива и выпуска отработанного

```
% definition of angular parameters of the rotor according to the motor phases
\def\Comp{0}
\def\Expl{360}
\def\Det{375}
\def\Ech{660}
\def\Asp{870}
\def\decalage{125}
% [en] shift the origin to position the rotor at the start of compression at time t = 0
```

Используется библиотека TikZ [39].

```
\begin{tikzpicture}
\colorlet{vertclair}{green!25}
\colorlet{grisclair}{gray!60}
\colorlet{rougepale}{red!60}
```

Различные стадии топливного цикла

```
% test needed to color the combustion chamber
\FPabs{\val}{\itheta}
% FPiflt and other FP tests do not do tests included in the tests. The first
% test is always true, the style will be affected with chambre1
% [{vertclair! \ Pos! Orange}] but will not be displayed if any of the
% following tests is true.
\FPiflt{\val}{1080}
\FPeval{\pos}{(\val-1080)/(\Ech-1080)*100}
```

```
\tikzstyle{chambre1}=[{vertclair!\pos!orange}]% aspiration
% \tikzstyle{chambre2}=[ ball color={gray!\pos!red}]% dA©tente}
% \tikzstyle{chambre3}=[ ball color={orange!\pos!green}]% aspiration
\def\texte{Aspiration}
\fi

% [fr] echappement / Second test, Exhaust
\FPiflt{\val}{\Asp}
\FPeval{\pos}{(\val-\Ech)/(\Asp-\Ech)*100}
\tikzstyle{chambre1}=[{vertclair!\pos!grisclair}]%echappement
\def\texte{Echappement}
\fi

% [fr] troisiA?me test - dA©tente  /  Third test - relaxation
\FPiflt{\val}{\Ech}
\FPeval{\pos}{(\val-\Det)/(\Ech-\Det)*100}
\tikzstyle{chambre1}=[ {grisclair!\pos!rougepale}]%dA©tente}
\def\texte{Detente}
\fi

% [fr] quatriA?me test - explosion /  Fourth test - explosion
\FPiflt{\val}{\Det}
\FPeval{\pos}{(\val-\Expl)/(\Det-\Expl)*100}
\tikzstyle{chambre1}=[ {rougepale!\pos!red}]%Explosion
\def\texte{Explosion}
\fi

% [fr] cinquiA?me test / Fifth test - aspiration
\FPiflt{\val}{\Expl}
\FPeval{\pos}{(\val-\Comp)/(\Expl-\Comp)*100}
\tikzstyle{chambre1}=[ {red!\pos!orange}]%compression
% \tikzstyle{chambre2}=[ ball color={gray!\pos!red}]%dA©tente}
% \tikzstyle{chambre3}=[ ball color={orange!\pos!green}]%aspiration
\def\texte{Compression}
\fi

\FPtrunc{\pos}{\pos}{0}

% [en] Adding the offset to draw the rotor in the first position
\FPeval{\itheta}{0-(\decalage+\itheta)}
\draw (-5,-4.5) rectangle (5,4.5); %cadre pour imposer les dimensions du dessin
```

Статор

```
%[fr}dessin du stator / drawing of the stator
```

```
\begin{scope}% stator
\coordinate (A) at (\itheta:\OA);  %[fr] Le point A tourne autour de O
% [fr] avec l'angle itheta
% [en]Point A turns around point O the angle with itheta
\coordinate (O) at (0,0);% Origine

\filldraw[thick,black,domain=0:1080,smooth,variable=\t,fill=gray,samples=50]
plot ({.4*cos(\t)+4*cos(.333333*\t)},{.4*sin(\t)+4*sin(.333333*\t)})
plot ({.42*cos(-\t)+4.2*cos(-.333333*\t)},{.42*sin(-\t)+4.2*sin(-.333333*\t)});
\fill[white](0.3,3) rectangle (0.5,4); % dessin des soupapes
% (admission echappement)
\fill[white](-0.3,3) rectangle (-0.5,4);
\filldraw[red](-0.4,-3.5) rectangle (-0.5,-3.7);% Bougie

% Coloring of the rooms according to the operating phase, the rotor will be
% plotted on top and hide the rest (the inner edge is straight, not curved).
% Chambre 1
\fill[thick,black,domain=-1*(\itheta):-1*(\itheta+360),smooth,variable=\t,
chambre1] plot ({.4*cos(-\t)+4*cos(-.333333*\t)},
{.4*sin(-\t)+4*sin(-.333333*\t)});

% affichage des paramA?tres
% \FPtrunc{\itheta}{\itheta}{0}
% \FPtrunc{\val}{\val}{0}
% \node at (0,6){$\theta$=\itheta,pos=\pos,val=\val};

\draw[black](O) circle (\OB); %Dessin du pignon fixe
\draw(O) -- (A);
\end{scope}
```

Рисование ротора

```
% [fr] Dessin du rotor / Draw the rotor
\begin{scope}[shift={(A)},rotate={\itheta}] % le repere est tournAⓒ de itheta
% [fr] les trois points, C, D, E sont dAⓒfinis en polaire dans ce repA?re
% [en] the three points, C, D, E are defined in the polar reference
\coordinate (E) at ({-\itheta*\OB/(\OB+\OA)}:\AE);
\coordinate (C) at ({-\itheta*\OB/(\OB+\OA)+120}:\AE);
\coordinate (D) at ({-\itheta*\OB/(\OB+\OA)+240}:\AE);
\draw[red](A) -- (E);
\filldraw [bend left=29.5,red,fill=red!50] (A) circle (\OA+\OB)
% dessin et coloriage du rotor
(E) to (D) to (C) to (E);
\end{scope}
```

```
\node at (0,-4) {\texte};  % le texte affiche la phase de fonctionnement
\end{tikzpicture}
}
```

Собственно LaTeX- документ.

```
\begin{document}
\begin{preview}
% [fr] Animation avec le package animate
% Animation with the animate package
\begin{animateinline}[controls,loop]{12}
\multiframe{72}{ixb=0+15}{
\Wankel{\ixb}}
\end{animateinline}
\vfill
%{fr] Pour dessiner le rotor dans un position particuliA?re
% To draw the rotor in a particular position
\begin{center}
\Wankel{200}
\end{center}
\end{preview}
\end{document}
```

■

6 Заключение

Система компьютерной верстки LaTeX, безусловно стала классической для создания научной литературы. Это уже часть общей культуры человечества. Система программирования векторной графики `METAPOST` - прекрасное средство сделать иллюстрации в едином стиле с основным текстом.

Безусловным ограничением `METAPOST` является отсутствие встроенной поддержки трехмерной графики. Пример 2.15 показывает большие трудозатраты при использовании встроенных собственных средств `METAPOST` . Наиболее естественным преемником является дескриптивный язык векторной графики `Asymptote` [38]. Помимо поддержки 3D, синтаксис в стиле C++, хорошую поддержку и интеграцию с наиболее популярным в настоящее время высокоуровневым языком программирования общего назначения Python. Следует упомянуть такие средства как PGF TikZ [39] и PSTricks [40]. Это список можно продолжить, но в одном из самых популярных приложений для LaTeX, TeXstudio [7], поддерживаются именно эти программные средства.

Уже само наличие большого количества последователей `METAPOST` говорит о том, что есть желание и потребность развивать идеи, заложенные в нем. Другое дело, что многие хорошие продукты написаны энтузиастами, развиваются медленно или уже фактически не поддерживаются. Время покажет, какие еще средства для LaTeXбудут эффективны и полезны.

Список литературы

[1] METAPOST. A User's Manual. John D. Hobby and the MetaPost development team. Documented version: 2.0rc2. February 19, 2018
https://www.tug.org/docs/metapost/mpman.pdf

[2] metapost – A development of METAFONT for creating graphics.
https://ctan.org/pkg/metapost

[3] С.М. Львовский. Набор и верстка в системе LaTeX. М. Издательство МЦНМО, 2014

[4] А.Баженов. Интервальный анализ. Основы теории и примеры применений. Часть 1. 2018
http://www.nsc.ru/interval/Education/Manuals/Bazhenov-InteAnalBasics.pdf

[5] А.Н. Швец. METAPOST. Краткий курс. 2009-12-18. Учебный материал для классов при механико-математическом факультете МГУ имени М.В.Ломоносова.
mech.math.msu.su/~shvetz/54/inf/metapost/mpshort.pdf

[6] Métapost : exemples
http://tex.loria.fr/prod-graph/zoonekynd/metapost/metapost.html

[7] TeXstudio. https://www.texstudio.org/, https://github.com/texstudio-org/texstudio/wiki

[8] Е.М. Балдин. Создание иллюстраций в MetaPost. (Версия 1.01). 2006.
http://www.inp.nsk.su/~baldin/mpost/mpillustration.pdf

[9] André Heck, Tutorial in MetaPost. 2003, ©AMSTEL Institute
http://tex.loria.fr/prod-graph/heck-metapost2003.pdf

[10] P. Grogono. Metapost: A Reference Manual. 22 June 2014
https://users.encs.concordia.ca/~grogono/Writings/mpref.pdf

[11] Asymptote: The Vector Graphics Language
http://asymptote.sourceforge.net/

[12] PyX — Python graphics package
http://pyx.sourceforge.net/

[13] PyX — Example: drawing/metapost.py.
Creating paths with MetaPost-like parameters
http://pyx.sourceforge.net/examples/drawing/metapost.html

[14] Deforming paths
http://nbviewer.jupyter.org/urls/sf.net/p/pyx/gallery/springs/attachment/springs.ipynb

[15] Paths with constant distance
http://nbviewer.jupyter.org/urls/sf.net/p/pyx/gallery/knot/attachment/knot.ipynb

[16] FEATPOST macros 2006
http://matagalatlante.org/nobre/featpost/doc/fpman.pdf
FEATPOST manual. L. Nobre G. 0.8.8
http://mirrors.ibiblio.org/CTAN/graphics/metapost/contrib/macros/featpost/doc/featpostmanual.pdf

[17] https://ctan.org/tex-archive/graphics/metapost/contrib/macros/featpost/example/cone.mp

[18] https://ctan.org/tex-archive/graphics/metapost/contrib/macros/featpost/example/torusexperiment.mp

[19] https://ctan.org/tex-archive/graphics/metapost/contrib/macros/featpost/example/twoDcolision.mp

[20] https://ctan.org/tex-archive/graphics/metapost/contrib/macros/featpost/example/xraycamera.mp

[21] https://ru.wikipedia.org/wiki/Камера_обскура
https://en.wikipedia.org/wiki/Camera_obscura

[22] https://ctan.org/tex-archive/graphics/metapost/contrib/macros/featpost/example/perspec.mp

[23] http://nuclphys.sinp.msu.ru/enc/e138.htm
https://en.wikipedia.org/wiki/Rutherford_scattering

[24] https://ctan.org/tex-archive/graphics/metapost/contrib/macros/featpost/example/antimattermeteor.mp

[25] https://ctan.org/tex-archive/graphics/metapost/contrib/macros/featpost/example/thethreekindsofperspec.mp

[26] http://ctan/graphics/metapost/contrib/macros/featpost/example/cubicfigures.mp

[27] https://ctan.org/tex-archive/graphics/metapost/contrib/macros/featpost/example/solardish.mp

[28] https://ctan.org/tex-archive/graphics/metapost/contrib/macros/featpost/example/simplelens.mp

[29] https://ru.wikipedia.org/wiki/Хрусталик
https://en.wikipedia.org/wiki/Lens_(anatomy)

[30] https://ctan.org/tex-archive/graphics/metapost/contrib/macros/
featpost/example/lgc2refractionEdited.mp

[31] https://ctan.org/tex-archive/graphics/metapost/contrib/macros/
featpost/example/torus.mp

[32] http://mirrors.mi.ras.ru/CTAN/graphics/metapost/contrib/macros/
featpost/example/geomfiguei.mp

[33] https://tex.stackexchange.com/questions/155395/
why-two-intersected-paths-cant-buildcycle-in-metapost

[34] https://tex.stackexchange.com/questions/345947/
how-to-get-the-point-of-intersection-of-an-arc-and-triangle-in-metapost

[35] http://ci.louisville.edu/tom/software/LaTeX/mpcirc/

[36] https://ctan.org/pkg/circuitikz

[37] http://www.texample.net/tikz/examples/area/electrical-engineering/

[38] https://en.wikipedia.org/wiki/Asymptote_vector_graphics_language

[39] https://en.wikipedia.org/wiki/PGF/TikZ

[40] https://en.wikibooks.org/wiki/LaTeX/PSTricks

[41] http://berndt-schwerdtfeger.de/wp-content/uploads/mp/sphere.mp

[42] https://tex.stackexchange.com/questions/385320/
how-to-use-metapost-to-draw-right-hand-rule-of-structural-mechanics

[43] http://tex.loria.fr/prod-graph/zoonekynd/metapost/metapost.html
exemeple 179

[44] http://ctan.altspu.ru/graphics/metapost/contrib/macros/featpost/
MPexamples.html

[45] https://en.wikipedia.org/wiki/Curve_of_constant_width

[46] http://www.texample.net/

[47] http://www.texample.net/media/tikz/examples/TEX/wankel-motor.tex

[48] https://en.wikipedia.org/wiki/Wankel_engine