«Подготовка геометрических чертежей с помощью языка MetaPost»

Часть 1. Планиметрия.

Московская гимназия на Юго-Западе №1543

Москва, 2006 г.

1. Введение

Скажем несколько слов о том, почему возникло желание написать такую работу. Учителю математики в наше время постоянно приходится готовить тексты с помощью компьютера. Эти тексты должны содержать помимо слов также формулы и чертежи. Повсеместная распространенность продуктов компании Microsoft привела к тому, что практически все пользуются для этого редакторами семейства Microsoft Word и Word-совместимыми надстройками. Этими средствами можно быстро и без особых интеллектуальных усилий получить хорошие результаты в области печати текста, редактор формул MathType несложен в изучении и даёт приемлемые, хотя далеко не блестящие результаты в плане печати математических формул, средства же для рисования чертежей оставляют желать много лучшего. Приходится либо избегать рисовать чертежи или чертить их от руки (как, к сожалению, поступает большинство коллег), либо осваивать специализированные графические приложения, такие как, например, "Живая геометрия", и встраивать созданные ими чертежи в документы Word, преодолевая традиционные для этого занятия трудности.

Мы рекомендуем для создания чертежей пользоваться системой MetaPost, представляющей собой специфический язык программирования, предназначенный для создания графических объектов. Освоение MetaPost'а требует некоторого времени, но это оправдывается возможностью создавать сложные чертежи высокого полиграфического качества. Особенностью, которая, как нам кажется, должна привлечь именно учителей математики к средствам MetaPost, является практически абсолютная независимость картинки-результата от богатства или бедности заложенных в систему шаблонов-возможностей, как это происходит в других системах. На экране Вы увидите только то и в точности то, что задали математически в файле-описании. Если Вы понимаете математическую природу каждой линии и точки на чертеже, Вы сможете создать совершенно точный графический образ, сколь угодно простой или сколь угодно сложный, ничего не "таская мышкой", не пользуясь "шаблонами" и т. п. Мы уверены, что знание MetaPost'а обогатит учителя математики, введёт в его арсенал мощное техническое средство, а работа с MetaPost'ом доставит интеллектуальное удовольствие. К сожалению, литературы, посвящённой MetaPost'у на русском языке практически нет (исключение составляет недавно появившаяся статья [2]). Автор надеется, что данная небольшая работа отчасти заполнит этот досадный пробел и выражает глубокую признательность своему коллеге, преподавателю информатики гимназии В. Д. Арнольду, за помощь в работе над данным текстом.

2. MetaPost — создание изображения.

Эта работа рассчитана на учителей математики — пользователей компьютера, но не профессиональных программистов, поэтому техническую часть мы будем излагать намеренно в предельно упрощённом виде. Нас будут интересовать только самые простые средства языка, позволяющие готовить геометрические чертежи — иллюстрации к задачам школьной планиметрии. Необходимое программное обеспечение мы будем считать уже установленным.

Именно, для операционной системы MsWindows версии 98 (и следующих) мы будем считать, что установлен пакет MikTeX (v. 2.2), программы Ghostscript (v. 7.05) и Ghostview (v. 4.3). Можно использовать и более поздние версии этих же продуктов, но в данном случае это не даёт существенного выигрыша. В операционных системах семейства UNIX, как правило, всё уже налажено. Но иногда приходится ставить пакет tetex и программу gv вместе с пакетом Ghostscript.

Все это можно взять (в соответствующих поддиректориях) с диска поставляемого вместе с

[3] или с сайтов CTAN (Comprehensive TeX archive network) —- www.ctan.org или ftp.chg.ru или tex.ihep.su, также можно воспользоваться сайтом МЦНМО (Московский Центр непрерывного математического образования) —ftp.mccme.ru/pub/tex/cd

Для набора текстов, задающих рисунки, понадобится текстовый редактор. Вообще говоря, любой. MsWord неудобен, Notepad лучше, очень удобен WinEdt, но он, в отличие от всего упоминавшегося ранее программного обеспечения, не является свободно распространяемым, то есть за лицензионную копию придётся платить.

В заявленной примитивной концепции будем считать, что установлены MetaPost и GsView, позволяющие, соответственно, создать (редактировать) и увидеть (распечатать) PostScript-файл — графический файл определённого формата. Для этого следует создать текстовый файл filename.mp, содержащий описание чертежа средствами MetaPost, применить к нему, собственно, сам MetaPost (командой mp filename.mp), результатом которой будет создание одного или нескольких файлов (например, `filename.1, filename.2,...`), просматриваемых программой gsview. Если Вы пользуетесь LaTeX'ом для создания текстов, то, добавив в преамбулу `\usepackage{epsfig}`, Вы сможете встроить картинку в Ваш текст, написав в требуемом месте `\epsfbox{filename.1}`. MetaPost, идеологически близкий к LaTex'у, содержит средства для вставки в рисунок набранных в LaTex'е формул и пр. Однако, самые простые необходимые подписи и обозначения можно добавлять и не владея LaTex'ом. Излагая приёмы построения чертежей в MetaPost'е, мы намеренно ограничимся описанием только простейших средств, отсылая желающих научиться большему к прекрасному руководству [1].

3. Рисуем квадрат

В этом и последующих разделах мы будем говорить о заполнении файла с расширением `.mp`, содержащего описание рисунка. Как увидеть сам рисунок, рассказано выше. Если Вы видите, что рисунок неудовлетворителен, Вы возвращаетесь к этому файлу (его обычно хназывают "исходником") и вносите в него необходимые изменения.

Итак, в файле picture.mp следует написать

```
beginfig(1);

endfig;
```

Между этими словами будет содержаться описание картинки. Перед `beginfig(1)` может быть ещё некоторый текст, называемый "преамбулой". Число в скобках — порядковый номер картинки, в одном файле их может быть описано несколько. В самом конце файла должно быть слово `end`.

Точка на рисунке задаётся парой чисел — своих координат. Начало координат не фиксировано. Масштаб, выбранный по умолчанию, не всегда устраивает, поэтому можно выбрать в преамбуле единицу длины, указав:

```
u=1cm;
```

При правильной настройке принтера, на печати отрезок длиной в 1u действительно будет около 1 см длиной. Если Вам захочется увеличить рисунок, измените эту строчку на

```
u=2cm;
```

и желаемый эффект произойдёт. Перед тем, как рисовать линии, нужно задать стиль линии. Их, стилей, весьма много, однако для наших простейших примеров достаточно написать так:

```
pickup pencircle scaled .8pt;
```

(смысл этой строки мы обсудим позже). Теперь можно нарисовать квадрат:

```
u=1cm;
beginfig(1);
pickup pencircle scaled.8pt;
draw(0,0)--(0,2u)--(2u,2u)--(2u,0)--(0,0);
endfig;
```

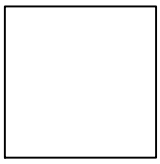

Рис. 1. Квадрат.

Покажем на этом примере некоторые конструкции языка MetaPost. Выражение типа `(2u,0)` — точка (тип `pair`). Запятая здесь отделяет абсциссу от ординаты. Десятичная запятая обозначается точкой. Начальный ноль можно опускать. Несколько точек, соединённых знаком `"--"` — конструкция, которая называется "путь" (`path`). Знак `"--"` указывает, что точки в пути соединяются отрезками. Команда `draw` рисует путь.

Для нас будут полезны ещё команды `label` и `dotlabel`. Они ставят около точки подпись, вторая отличается от первой тем, что саму точку рисует жирной, так, как на чертежах принято обозначать вершины многоугольников и пр. Расположение подписи относительно подписываемой точки определяет "суффикс", следующий через точку после имени команды. Суффиксов восемь:

`ulft`(сверху-слева)	`top` (сверху)	`urt` (сверху-справа)
`lft` (слева)	**ТОЧКА**	`rt` (справа)
`llft`(снизу-слева)	`bot` (снизу)	`lrt` (снизу-справа)

Нарисуем теперь квадрат $ABCD$:

```
beginfig(2);
pickup pencircle scaled .8pt;
draw(0,0)--(0,2u)--(2u,2u)--(2u,0)--cycle;
dotlabel.llft("A",(0,0));
dotlabel.ulft("B",(0,2u));
dotlabel.urt("C",(2u,2u));
dotlabel.lrt("D",(2u,0));
endfig;
```

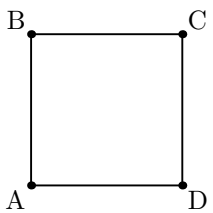

Рис. 2. Размеченный квадрат.

Здесь мы помимо добавления отмеченных вершин заменили последний переход к вершине A на слово `cycle`. Это — указание MetaPost'у на то, что следует вернуться к началу пути, и что путь при этом замкнут. Этим можно воспользоваться для того, чтобы закрасить внутренность пути.

```
beginfig(3);
path p;
pickup pencircle scaled .8pt;
P:=(0,0)--(0,2u)--(2u,2u)--(2u,0)--cycle;
draw p;
dotlabel.llft("A",(0,0));
dotlabel.ulft("B",(0,2u));
dotlabel.urt("C",(2u,2u));
dotlabel.lrt("D",(2u,0));
fill p withcolor red;
endfig;
```

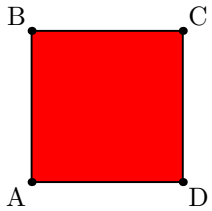

Рис. 3. Красный квадрат (по Малевичу).

В этом примере обратите внимание на введение переменной `p` типа `path`, присваивание ей с помощью оператора присваивания `":="` некоторого значения и последующим рисованием пути (`draw p`) и «заливанием его краской» командой с очень прозрачной структурой `fill p withcolor red`. Имена переменных различных типов (точка, путь и другие) строятся по определённым правилам, но для наших целей достаточно знать, что именем может быть последовательность латинских букв (размер букв интерпретатор языка различает!). Имя должно быть определено в начале описания как имя переменной данного типа.

Научившись рисовать красный квадрат, перейдём к более содержательным чертежам. Для примеров в этой работе были взяты некоторые задачи из заочного тура второй олимпиады по геометрии имени И. Ф. Шарыгина (весна 2006 г.).

4. Аффинные структуры

MetaPost предоставляет большие возможности для рисования прямых линий. Про точки плоскости (тип данных `pair`) уже говорилось. Та же точка может пониматься как радиус-вектор, поэтому точки можно складывать, вычитать и умножать на числа. Так, например, можно посчитать `(2, -3)+2.5(8,2)=(22,2)` и т.п. Соответственно прямая задаётся своим параметрическим уравнением: прямая, проходящая через точки `a` и `b` состоит из точек вида `a+k(b-a)`, где `k` — переменная типа «вещественное число» (тип `numeric`). Для этой часто встречающейся конструкции есть упрощённая запись с помощью квадратных скобок: вместо `c:=a+0.5(b-a)` можно написать `c:=0.5[a,b]` или `c=0.5[a,b]`. (Естественно, обе эти записи определяют `c` как середину отрезка `[a,b]`.) Между последними двумя записями есть существенная разница. Если в первой записи переменной `c` присваивается новое значение, то вторая запись представляет собой уравнение с неизвестной `c`, которое решается MetaPost'ом, и переменной присваивается вычисленное значение. Если таких уравнений (линейных) написать несколько, то MetaPost пытается решить систему написанных уравнений и определить значения переменных. Если это невозможно (например, когда написано `a=(2,3); b=2a; c=0.5[a,b]; a+c=b;`)при компиляции обозначится ошибка. Пересечение прямых линий требуется находить очень часто, для этого существует специальная запись: `e=whatever[a,b]=whatever[c,d]`. Здесь точка `e` вычислена как пересечение прямых `[a,b]` и `[c,d]`, а под `whatever[a,b]` понимается произвольная точка прямой `[a,b]`. Конечно же и здесь происходит не присваивание, а решение системы уравнений; употребление знака `:=` привело бы к ошибке.

Рассмотрим для примера задачу №13 олимпиады (автор А. Акопян).

> Даны две прямые a и b, а также точки A и B. Точка X скользит по прямой a, а точка Y по прямой b, так что $AX \parallel BY$. Найдите ГМТ пересечения AY с XB.

Прямые, конечно, могут пересекаться и быть параллельными. Рассмотрим общий случай пересекающихся прямых. Попробуем нарисовать картинку. Пусть $AX \cap BY = Q$.

```
beginfig(4);
pair iq,o,p,q,r,s,a,b,ix,iy;
pickup pencircle scaled .8pt;
p:=(-3u,0);
q:=(9u,0);
r:=(-u,-2u);
s:=(3u,6u);
a:=(5u,2u);
b:=(6u,u);
ix:=(.3u,.6u);
z=b+ix-a;
iy=whatever[z,b]=whatever[p,q];
draw p--q;
draw r--s;
label.bot("b", (8.7u,0));
label.lrt("a",(3u,6u));
o=whatever[p,q]=whatever[r,s];
label.llft("O", o-(.2u,0));
dotlabel("", o);
dotlabel.urt("A",a);
dotlabel.rt("B",b);
dotlabel.lft("X",ix);
dotlabel.bot("Y",iy);
draw a--ix;
draw b--iy;
draw b--ix dashed evenly;
draw a--iy dashed evenly;
iq=whatever[b,ix]=whatever[a,iy];
dotlabel.bot("Q",iq);
endfig;
```

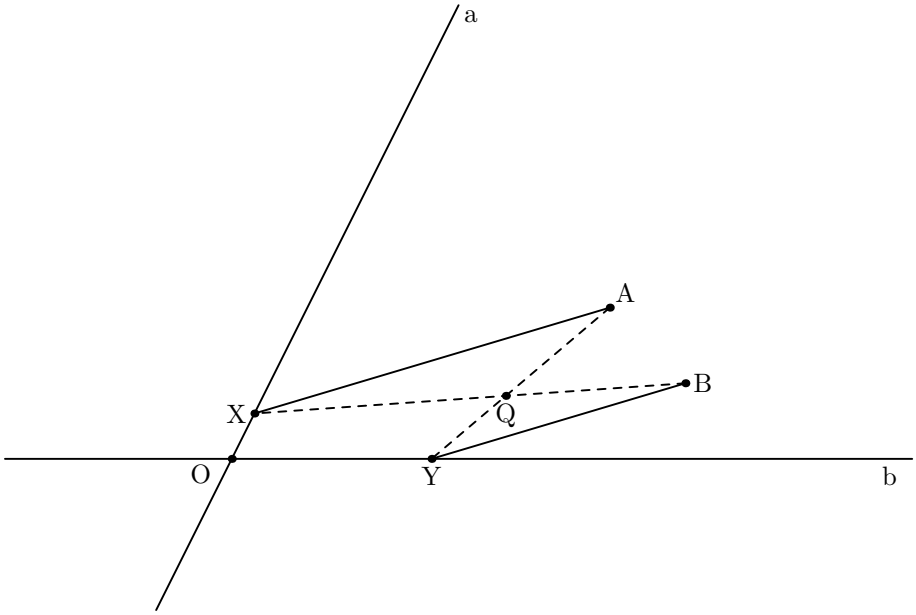

Рис. 4. Чертёж к задаче №13.

Прокомментируем код картинки. Он уже достаточно длинный. Команды отделяются точкой с запятой и могли бы быть записаны в строчки более экономно; мы же поместили (и будем помещать в дальнейшем) каждую команду на отдельной строке для более удобного чтения. Пробелы интерпретатор языка игнорирует. Вначале определено множество переменных типа «точка» (`pair`). Они поименованы в основном как точки в задаче; точки `p`, `q` и `r`, `s` задают условные «концы» прямых b и a на чертеже (первая прямая выбрана горизонтальной, вторая под углом $\operatorname{arctg} 2$ к ней). За точку X отвечает переменная `ix`, произвольно выбранная на прямой a (в данном случае она выбрана "руками"). Далее, запись `z=b+ix-a` определяет точку Z так, что $\overrightarrow{BZ} = \overrightarrow{AX}$. Тем самым прямая BZ получается параллельной к прямой AX, и мы можем построить точку Y (`iy=whatever[z,b]=whatever[p,q]`). После этого определяется точка Q. Прочие

строчки кода посвящены рисованию линий и точек (обратите внимание, что некоторые точки мы вводим, но не рисуем); отметим ещё `label.llft("O",o-(.2u,0))` — буква O несколько налезала на прямые, поэтому точку, слева внизу от которой она подписана, пришлось чуть сместить. И последний комментарий: `draw b--ix dashed evenly` рисует пунктирную линию вместо сплошной.

Пока непонятно, правда, как решить задачу, но, оказывается, MetaPost может помочь и в этом! Впоследствии мы ещё вернёмся к ней.

5. Трансформации

Точки и пути можно подвергать разнообразным преобразованиям, из которых наиболее употребительными будут движения — сдвиг, поворот, осевая симметрия, а также гомотетия. Результаты применения к точке `z` сдвига на вектор `b`, поворота на угол $\varphi = n^\circ$ вокруг начала координат, осевой симметрии относительно оси абсцисс и гомотетии с коэффициентом k относительно начала координат записываются соответственно так:

```
z shifted b;   z rotated n;   z xscaled (-1);   z scaled k;
```

Понятно, что `xscaled` умножает абсциссу точки на соответствующее число, что в общем случае означает сжатие к оси, а для осуществления симметрии мы выбрали число -1. Теперь ясно слово `scaled` в загадочной строчке `pickup pencircle scaled .8pt` задаёт толщину линии в «пунктах» — особых единицах длины. Попробуем нарисовать чертёж к ещё одной задаче.

Задача №4б олимпиады (автор И. Богданов).

> Даны два правильных пятиугольника, $OKLMN$ и $OPRST$, причём точка P лежит на отрезке ON, а точки R, S, T — вне пятиугольника $OKLMN$. Найдите угол между прямыми KP и MS.

Первый пятиугольник построим с центром в начале координат, второй же зададим как образ первого при поворотной гомотетии с центром O и углом 108°.

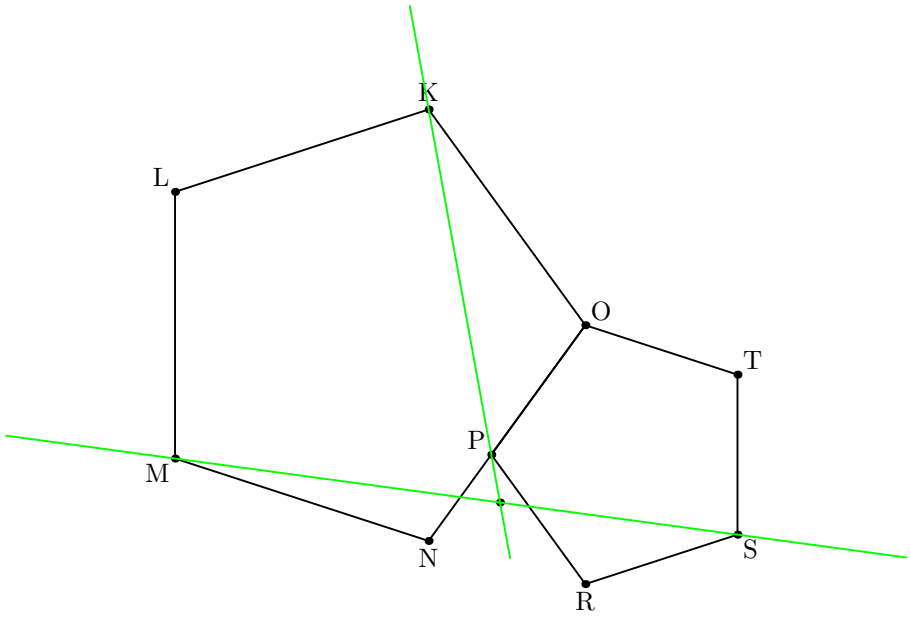

Рис. 5. Чертёж к задаче №4б.

По картинке видно, что этот угол больше 60°, так что, исходя из характерных величин углов в данной задаче, можно предположить, что ответ — 72°. Если это так, то точка пересе-

чения прямых лежит на описанных окружностях обоих пятиугольников. Добавим окружности к чертежу. Вроде бы так оно и есть:

```
beginfig(6);
pickup pencircle scaled .8pt;
pair o,k,l,m,n,p,r,s,t,q;
path kr,krr;
o:=(3u,0);
k=o rotated 72;
l=k rotated 72;
m=l rotated 72;
n=m rotated 72;
p=k shifted -o scaled 0.6 rotated 108 shifted o;
r=l shifted -o scaled 0.6 rotated 108 shifted o;
s=m shifted -o scaled 0.6 rotated 108 shifted o;
t=n shifted -o scaled 0.6 rotated 108 shifted o;
draw o--k--l--m--n--cycle;
draw o--p--r--s--t--cycle;
dotlabel.urt("O",o);
dotlabel.top("K",k);
dotlabel.ulft("L",l);
dotlabel.llft("M",m);
dotlabel.bot("N",n);
dotlabel.ulft("P",p);
dotlabel.bot("R",r);
dotlabel.lrt("S",s);
dotlabel.urt("T",t);
q=whatever[k,p]=whatever[m,s];
dotlabel.llft("Q",q);
draw k-.3(p-k)--k+1.3(p-k) withcolor green;
draw m-.3(s-m)--m+1.3(s-m) withcolor green;
pickup pencircle scaled .4pt;
kr:=fullcircle scaled 6u;
krr:=fullcircle scaled 6u shifted -o scaled 0.6 rotated 108 shifted o;
draw kr withcolor blue;
draw krr withcolor blue;
endfig;
```

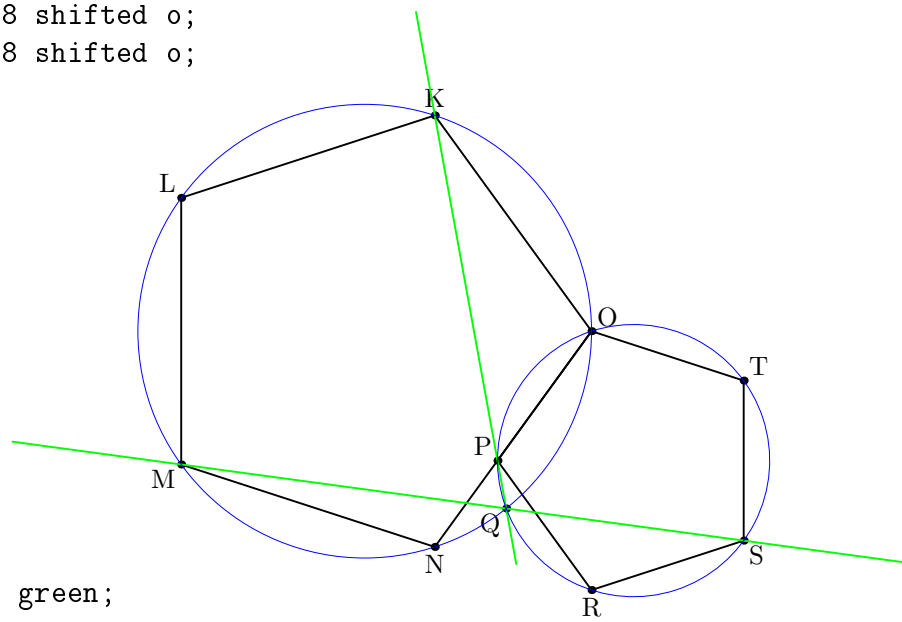

Рис. 6. Чертёж-подсказка к задаче №4б.

В качестве комментариев нужно в первую очередь отметить, что `fullcircle scaled 6u` определяет путь, являющийся окружностью с центром в начале координат и **диаметром** `6u`. Обе окружности описаны в начале кода как пути. Сложное движение

```
shifted -o scaled 0.6 rotated 108 shifted o
```

представляет собой поворотную гомотетию вокруг точки `o`, сдвиги нужны для того, чтобы свести движение к началу координат и, после выполнения, вернуть обратно.

6. Использование макросов

MetaPost позволяет создать макрос, то есть однократно описать часто встречающееся действие по определённым правилам, дать ему имя и впоследствии писать это имя вместо полного описания всей процедуры. Также можно спрятать в макрос вычисление какого-то параметра, и тогда с именем макроса можно оперировать, как с указанным параметром. Правила создания макросов не слишком просты и в этой работе описаны не будут. Однако ничто не мешает каждому пользоваться библиотеками уже написанных макросов. Многие макросы, написанные как автором работы, так и другими людьми, облегчают создание чертежей и находятся в свободном доступе в локальной сети гимназии. Приведём пример создания чертежа с использованием макросов.

Задача №19 олимпиады (автор Л. Емельянов).

> Через середины сторон треугольника T проведены прямые, перпендикулярные биссектрисам противолежащих углов треугольника. Эти прямые образовали треугольник T_1. Докажите, что центр описанной около T_1 окружности находится в середине отрезка, образованного центром вписанной окружности и точкой пересечения высот треугольника T.

Видно, что в условии задачи фигурируют часто встречающиеся элементы геометрии треугольника: биссектриса, ортоцентр и др. Для их нахождения будут использованы макросы.

```
beginfig(7);
pickup pencircle scaled .4pt;
pair a,b,c,al,bl,cl,am,bm,cm;
pair ah,bh,ch,p,q,r,o,h,i;
a:=(4u,0);
b:=(u,3u);
c:=(-2u,-u);
al:=osn_biss(a,b,c);
bl:=osn_biss(b,c,a);
cl:=osn_biss(c,a,b);
am:=.5[b,c];
bm:=.5[a,c];
cm:=.5[b,a];
ah:=proection(am,a,al);
bh:=proection(bm,b,bl);
ch:=proection(cm,c,cl);
p=whatever[am,ah]=whatever[bm,bh];
q=whatever[am,ah]=whatever[cm,ch];
r=whatever[cm,ch]=whatever[bm,bh];
o:=op_centre(p,q,r);
h:=ortocentre(a,b,c);
i:=incentre(a,b,c);
draw a--b--c--cycle;
draw p-.1(q-p)--p+1.1(q-p) withcolor blue;
draw q-.1(r-q)--q+1.1(r-q) withcolor blue;
draw r-.1(p-r)--r+1.1(p-r) withcolor blue;
```

```
draw i--o--h withcolor red;
dotlabel("",a);
dotlabel("",b);
dotlabel("",c);
dotlabel("",p);
dotlabel("",q);
dotlabel("",r);
pickup pencircle scaled 4pt;
draw i--i;
draw o--o;
draw h--h;
endfig;
```

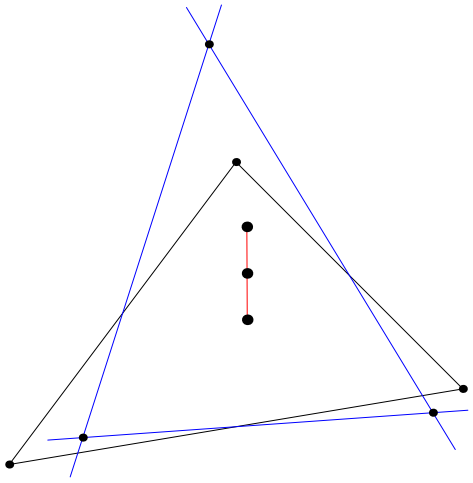

Рис. 7. Чертёж к задаче №19.

Здесь `osn_biss(a,b,c)` — точка, служащая основанием биссектрисы, проведённой из вершины `a` треугольника `abc`, `incentre(a,b,c)` — центр вписанной окружности треугольника `abc` и так далее. Отметим ещё такой приём: желая сделать три точки жирнее обычных, мы рисуем их как линию, у которой начало и конец совпадают, а толщину линии делаем большой.

Отметим, что утверждение задачи действительно представляется верным. Мы не специально ставили точки так, чтобы одна оказалась посередине — такое расположение получилось для вычисленных аналитически точек. Однако хорошо, когда положение всех точек ясно. Иногда бывает и не так. Рассмотрим пример задачи.

Задача №7 олимпиады (автор Д. Калинин).

> Внутри квадрата $ABCD$ взята точка E, а вне — точка F, так что треугольники ABE и CBF равны. Найдите углы треугольника ABE, если известно, что отрезок EF равен стороне квадрата, а угол BFD — прямой.

Тут положение точек не ясно изначально. Попытаемся нарисовать чертёж «на глазок». Точку E выберем на диагонали, вдвое ближе к A, чем к C.

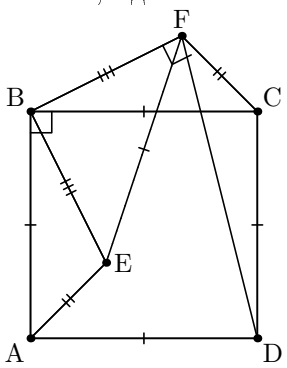

Рис. 8. Чертёж к задаче №7. Первая попытка.

Видно, что вышло не очень. Макрос, рисующий значок прямого угла, явно делает что-то не то. Попробуем понять задачу точнее. Пусть Q — центр квадрата. Тогда F лежит на описанной окружности квадрата с центром Q. Из равенства треугольников ABE и CBF ясно, что $\angle ABE = \angle CBF$ и $\angle EBF = 90°$, так что выходит, что треугольник BFQ правильный. Значит, можно точно нарисовать картинку. Вот вторая попытка:

```
beginfig(9);
pair a,b,c,d,e,f,q;
pickup pencircle scaled .7pt;
a:=(0,0);
b:=(0,3u);
c:=(3u,3u);
d:=(3u,0);
q:=.5[a,c];
f:=q shifted -b rotated 60 shifted b;
e:=q shifted -b rotated -30 shifted b;
draw a--b--c--d--a--e--b--f--c;
draw f--d;
dotlabel.llft("A", a);
dotlabel.ulft("B", b);
dotlabel.urt("C", c);
dotlabel.lrt("D", d);
dotlabel.rt("E", e);
dotlabel.top("F", f);
draw e--f dashed withdots;
mark_rt_angle(a, b, c);
mark_rt_angle(b, f, d);
draw_marked(a--b, 1);
draw_marked(b--c, 1);
draw_marked(c--d, 1);
draw_marked(d--a, 1);
draw_marked(e--f, 1);
draw_marked(a--e, 2);
draw_marked(c--f, 2);
draw_marked(e--b, 3);
draw_marked(b--f, 3);
endfig;
```

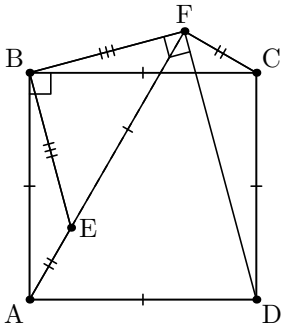

Рис. 9. Чертёж к задаче №7. Вторая попытка.

Теперь всё правильно. Фактически мы уже решили задачу. $\angle QBF = 60°$, $\angle QBC = 45°$, поэтому $\angle CBF = 15°$. $\angle BCF = \frac{1}{2}\angle BQF$, как вписанный, то есть $\angle BCF = 30°$. Тогда $\angle BFC = 135°$. Обратим внимание на команду, которая рисует линию точками. Значки углов и штрихи, отмечающие равные отрезки, нарисованы макросами.

Между прочим, на чертеже явно обнаружилось, что E лежит на AF, что само по себе интересно и может быть вопросом задачи. Любопытно, что точность MetaPost'а здесь отчасти может «пойти во вред» интересам учителя — MetaPost «не даёт»нарисовать неправильную картинку и приходится верным чертежом подсказывать решающему. Впрочем, можно и первый чертёж сделать правдоподобным, хотя это и сложнее.

7. Вычислительный эксперимент

Мы можем попробовать применить MetaPost для проведения геометрического построения с целью проверки какого-то предположения или накопления данных. Один раз мы уже делали

это при обсуждении задачи №4б, когда экспериментально подтвердили, что прямые и окружности пересекаются в одной точке. Теперь попытаемся решить задачу №13 про ГМТ точек пересечения.

Проведём эксперимент. Запустим несколько положений X, по ним построим Y и Q. Точки Q отметим, а все построения скроем, а то будет паутина линий.

```
beginfig(10);
pair iq,o,p,q,r,s,a,b,ix,iy,z;
pickup pencircle scaled .8pt;
p:=(-3u,0);
q:=(9u,0);
r:=(-u,-2u);
s:=(3u,6u);
a:=(5u,2u);
b:=(6u,u);
draw p--q;
draw r--s;
label.bot("b", (8.7u,0));
label.lrt("a",(3u,6u));
o=whatever[p,q]=whatever[r,s];
label.llft("O", o-(.2u,0));
dotlabel("", o);
dotlabel.urt("A",a);
dotlabel.rt("B",b);
pickup pencircle scaled 2.8pt;
for i=1 upto 15:
ix:=r+.025*i*(s-r);
z:=b+ix-a;
iy:=whatever[z,b]=whatever[p,q];
iq:=whatever[b,ix]=whatever[a,iy];
draw iq--iq withcolor blue;
endfor;
for i=29 upto 40:
ix:=r+.025*i*(s-r);
z:=b+ix-a;
iy:=whatever[z,b]=whatever[p,q];
iq:=whatever[b,ix]=whatever[a,iy];
draw iq--iq withcolor blue;
endfor;
endfig;
```

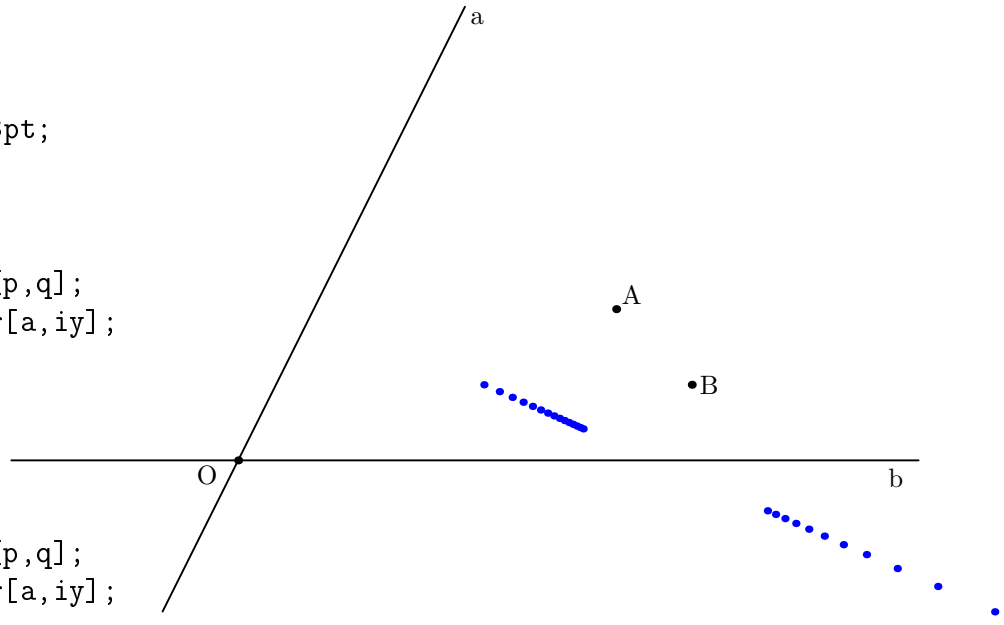

Рис. 10. Вычислительный эксперимент к задаче №7.

Новой для нас в этом описании будет структура, начинающаяся с `for`: это типичная для языков программирования организация цикла. Запись понятна из примера: после `for` переменная уравнивается с начальным целым значением, дальше слово `upto` и конечное целое значение, не меньшее начального (меньшее тоже можно, но тогда требуется писать `downto`), потом двоеточие. После двоеточия набор команд, которые будут повторены для всех значений переменной. Набор ограничивается словом `endfor`, и весь текст между `for` и `endfor` понимается, как одна

составная команда. В приведённом примере цикл пришлось разбить, так как для некоторого диапазона `i` точки `iy` «уползали» слишком далеко, и происходила вычислительная ошибка.

Результат эксперимента ясен — искомое ГМТ является прямой линией (или во всяком случае, её частью). Проведём эту прямую и посмотрим на неё внимательно.

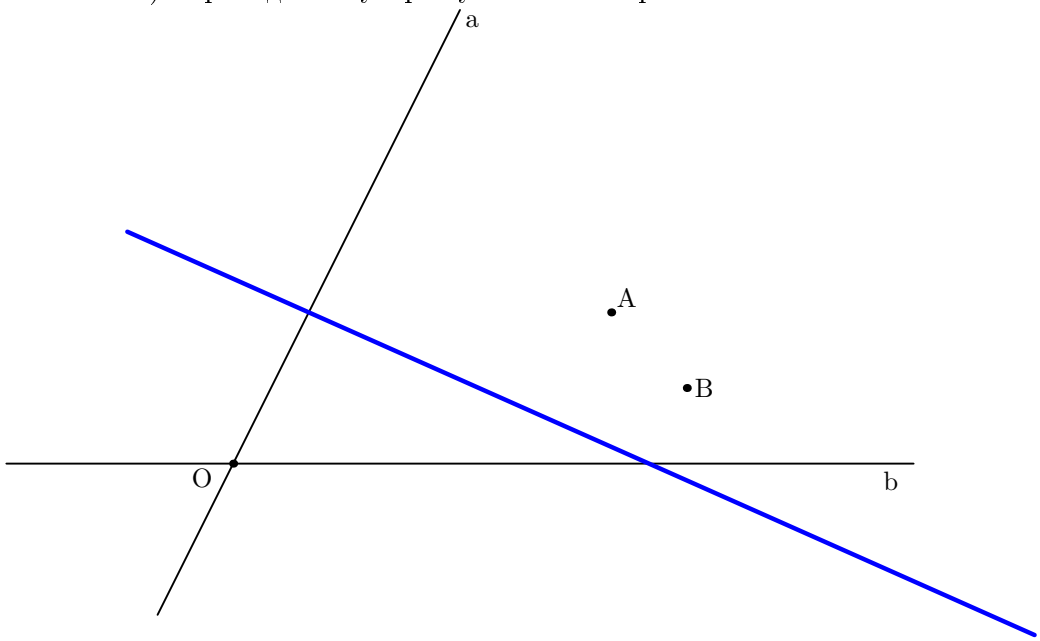

Рис. 11. Предполагаемый ответ в задаче №7.

Понаблюдав, понимаем, что точки пересечения синей прямой с осями суть проекции A и B на одноимённые оси. Назовём их U и V соответственно и наложим самую первую картинку (рис.4) на только что полученную. Получается вот что:

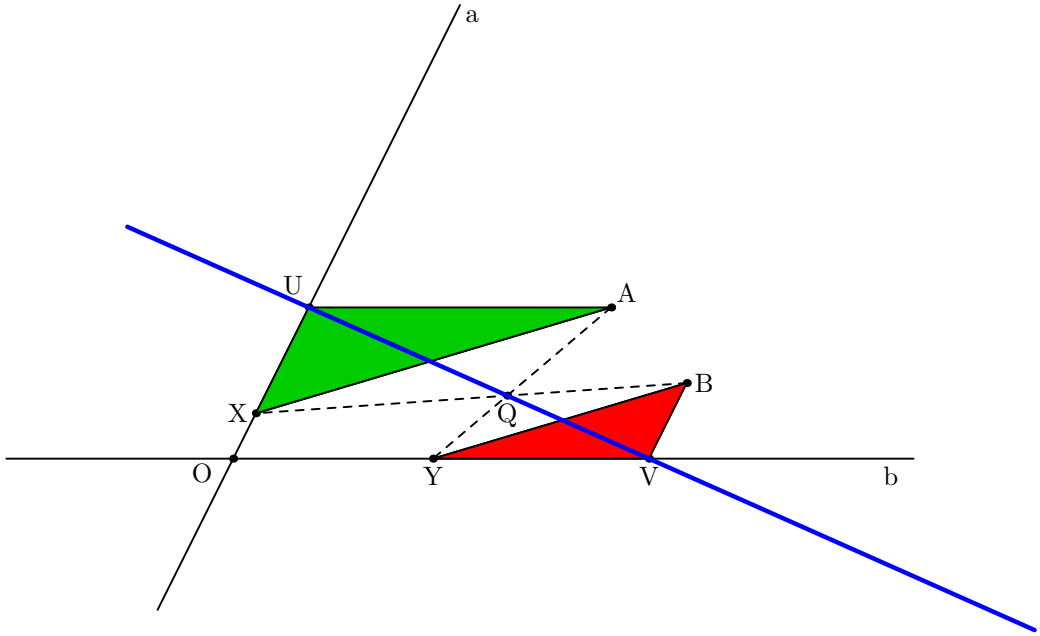

Рис. 12. Чертёж-решение к задаче №7.

Задача фактически решена: ясно, что цветные треугольники гомотетичны, Q — центр гомотетии, точки U и V, в частности, тоже гомотетичны, поэтому Q лежит на найденной прямой. Ясно, что для любой точки прямой такие треугольники можно построить.

8. Заключение

Повторим очень кратко основные факты, изложенные в этой небольшой работе. Попробуем остановиться на самом важном, а также акцентировать внимание на том, что прозвучало вскользь. Итак...

Каждый чертёж задаётся текстом, расположенным между `beginfig(<номер чертежа>);` и `endfig;`. Команды отделяются друг от друга точкой с запятой. Пробелы игнорируются.

В программе приходится оперировать с переменными, которые бывают различных типов. Имена переменных — последовательности латинских букв и цифр (но начинаться должны с букв), размер букв существенен. Всего типов девять, но мы пока столкнулись с наиболее распространёнными: `numeric` — вещественные числа, `pair` — точки на плоскости, задаваемые как пар чисел, и `path` — пути, то есть последовательности точек, соединяемые в линию определённым образом. Будет полезно знать, что с координатами точки можно работать по отдельности: если `a` — переменная типа `pair`, то `xpart a` и `ypart a` суть абсцисса и ордината этой точки (обе, разумеется, типа `numeric`). Мы использовали только пути-ломаные, однако возможны и криволинейные пути. Для этого вместо `"--"` в описании пути следует писать `". ."`. Тогда точки будут соединяться плавной линией, представляющей собой кубическую кривую, так называемый сплайн Безье. Его параметрами можно управлять, но описание средств такого управления сильно уведёт нас от основной темы.

Переменной присваивается значение с помощью оператора присваивания `":="`. С числами можно производить все четыре арифметические операции, точки складывать, вычитать и умножать на числа. Разумеется, есть ещё много операций, например `a dotprod b` есть скалярное произведение векторов `a` и `b` и так далее. На переменные можно написать несколько линейных уравнений, и значения переменных будут найдены из предъявленной системы, если это, конечно, возможно.

Рисование пути производится командой `draw`. Добавление `dashed evenly` к команде рисования пути приводит к его изображению пунктирной линией, а добавление `dashed withdots` — к рисованию пути точками. Окружность с центром в начале координат и диаметром `d` задаётся командой `fullcircle scaled d`. Заполнение внутренности пути проводится командой `fill`. Рисование и подписывание точек командами `dotlabel` или `label` с соответствующими суффиксами и параметрами в виде текста подписи и координаты точки. Повторим ещё раз таблицу суффиксов.

`ulft`(сверху-слева)	`top` (сверху)	`urt` (сверху-справа)
`lft` (слева)	**ТОЧКА**	`rt` (справа)
`llft`(снизу-слева)	`bot` (снизу)	`lrt` (снизу-справа)

К путям и точкам применимы трансформации, например:

`z shifted b` сдвигает `z` на векторb

`z rotated n` поворачивает `z` на угол $\varphi = n^\circ$ вокруг начала координат

`z xscaled k` удаляет `z` в `k` раз от оси абсцисс

`z yxscaled k` удаляет `z` в `k` раз от оси ординат

`z scaled k` удаляет `z` в `k` раз от начала координат

В коде картинки возможно организовать цикл. Для этого пишется

`for i=m upto n: <набор команд> endfor;`. Здесь набор команд будет повторён для всех значений `i` от `m` до `n`.

Литература

[1] John D. Hobby «A User's manual for MetaPost». — Документация к пакету MetaPost на http://www.ctan.org/tex-archive/graphics/metapost/

[2] Балдин Е.М. «Создание иллюстраций в MetaPost». Linux Format, №6, 2006 (http://www.linuxformat.ru)

[3] Львовский С. М. «Набор и верстка в пакете LaTeX» (3 изд., М.: МЦНМО, 2003).