LaTeX, GNU/Linux и русский стиль.

© Е.М. Балдин*

Эта статья была опубликована в декабрьском номере русскоязычного журнала Linux Format (`http://www.linuxformat.ru`) за 2006 год. Статья размещена с разрешения редакции журнала на сайте `http://www.inp.nsk.su/~baldin/` и до мая месяца все вопросы с размещением статьи в других местах следует решать с редакцией Linux Format. Затем все права на текст возвращаются ко мне.

Текст, представленный здесь, не является точной копией статьи в журнале. Текущий текст в отличии от журнального варианта корректор не просматривал. Все вопросы по содержанию, а так же замечания и предложения следует задавать мне по электронной почте `mailto:E.M.Baldin@inp.nsk.su`.

Текст на текущий момент является просто *текстом*, а не книгой. Поэтому результирующая доводка в целях улучшения восприятия текста не проводилась.

*e-mail: E.M.Baldin@inp.nsk.su

Эмблемы TeX и METAFONT, созданные Дуайном Бибби, взяты со странички Д.Э. Кнута. Цветной пингвин взят из пакета `ps2pdf` от Ральфа Найпрашека (Rolf Niepraschk)

Оглавление

Глава 4

Графика

Q. Как быстро написать курсовую на LaTeX'е в которой кроме текста есть и графики?
A. Сделать для начала графики ☺

Вопрос и краткий ответ.

Вероятно, TeX на текущий момент лучше других программ вёрстки умеет разбивать абзацы на строки. То есть удачнее всех разливать порции «клея» между «боксами», но подготовка графики выносится за рамки этого процесса. Почти ...

С точки зрения TeX картинка — это просто очень большой прямоугольник, который надо как-то разместить на странице. От пользователя нужны только размеры этого прямоугольника. Отображение же иллюстрации лежит на плечах драйверов. Самым востребованным форматом для представления графики в LaTeX до сих пор является Encapsulated PostScript.

4.1. Encupsulated PostScript

Уже больше двадцати лет прошло с тех пор как никому не известная фирма Adobe Systems получила инвестиции от фирмы Apple на «обучение» лазерных принтеров молодому языку PostScript. Как следствие этот платформонезависимый язык с полностью открытой спецификацией стал безальтернативным стандартом. Даже сейчас PostScript фактически не имеет конкурентов в области допечатной подготовки. Поэтому почти все «уважающие себя» графические программы умеют экспортировать результаты своей деятельности в виде инструкций PostScript. Особенно это касается векторных графических редакторов, так как PostScript подразумевает векторную графику.

Encapsulated PostScript или кратко EPS — графический формат. Файлы в этом формате обычно имеют расширение `.eps`. По сути дела это PostScript с некоторыми упрощениями и дополнительными договорённостями. Самая интересная с точки

зрения LaTeX договорённость — это обязательное наличие в заголовке информации о размере картинки, которая передаётся вместе с комментарием:

```
%!PS-Adobe-2.0 EPSF-2.0
%%Creator: dvips(k) 5.95b Copyright 2005 Radical Eye Software
%%Title: picture.dvi
%%BoundingBox: 127 464 430 667
%%DocumentFonts: SFRM1200 SFRM0800
%%EndComments
```

Первая строка комментария, обычно, содержит версию PostScript[1]. Вслед за комментарием `BoundingBox` идёт информация о размерах. Первые два числа соответствуют координатам левого нижнего угла картинки, а последние соответствуют координатам правого верхнего угла. Единицей измерения является «большой пункт» (bp=1/72 in), который примерно равен 0.351 мм. Для вёрстки текста указанной информации достаточно.

Чтобы из уже имеющегося одностраничного PostScript-файла сделать EPS необходимо и, как правило, достаточно добавить `BoundingBox`. Для вычисления искомых размеров можно воспользоваться утилитой `ps2eps` из одноимённого пакета. Если же в стандартной поставке эта программа отсутствует, то можно напрямую воспользоваться программой Ghostscript — свободным программным интерпретатором PostScript:

```
> gs -q -dSAFER -dNOPAUSE -dBATCH -sDEVICE=bbox «имя файла»
```

Размеры выясняются с помощью указания специального драйвера `bbox`. Ключи `-q`, `-dNOPAUSE` и `-dBATCH` используются для подавления не нужной информации и вопросов со стороны программы. Ключ `-dSAFER` гарантирует что Ghostscript не будет производить никаких деструктивных действий[2].

Ещё одной особенностью EPS-формата является возможность добавлять растровое изображения для предварительного просмотра. Это было сделано для случаев, когда программы не понимают PostScript, но что-то на месте дырки для картинки отобразить надо. Такое добавление идёт в разрез с принципиальной кроссплатформенностью PostScript и по возможности добавление картинки для предварительного просмотра следует избегать. Но в любом случае для операции с этим расширением, в том числе и для добавления/удаления можно воспользоваться утилитой `epstool` из одноимённого пакета.

В конце рассказа про EPS хотелось бы упомянуть о замечательной утилите `pstoedit` из, естественно, одноимённого же пакета. Не все, но некоторые из более-менее внятно созданных PostScript-файлов она ухитряется перевести в редактируемый век-

[1] Некоторые программы перед комментарием добавляют бинарный мусор. Не будем тыкать пальцем в драйвер вывода для PostScript-принтеров одной очень распространённой альтернативной операционной системы. Для полноценной работы с такими файлами этот мусор необходимо удалить.

[2] Отключается возможность выполнения таких команд, как удаление и переименование, а чтение файлов происходит в режиме read-only. Очень полезный ключ, если Ghostscript используется в качестве фильтра.

торный графический формат. Это упрощает работу с правкой файлов, которые не имеют исходников.

4.2. Как из растра сделать EPS

Одним из важных вопросов является конвертация растровых форматов в EPS. Растр гораздо проще создавать. Кое-где, например в случае снимков экрана, применение растра оптимальнее векторных форматов. Стандартные подходы, как в случае утилиты `convert` из пакета ImageMagick, не всегда дают оптимальные результаты.

Возможным и вполне разумным решением является замена традиционной линейки: `latex`→`dvips`→`[ps2pdf]` на `pdflatex`, который сразу из коробки поддерживает растровые форматы PNG[3] и JPEG[4] которые можно внедрять в формат PDF[5] на прямую. Массового перехода на эту технологию пока не видно, но заметное движение в эту сторону есть. У неё есть неоспоримые достоинства, но она не лишена недостатков. Рассказ о `pdflatex` выходит за рамки *этой* статьи.

Конвертацию из JPEG можно решить с помощью простой утилитки `jpeg2ps`, которую можно найти на любом CTAN[6]-архиве в директории `nonfree/support/jpeg2ps`. Утилита не преобразовывает JPEG-файл, а просто добавляет правильный eps-заголовок. Декомпрессия JPEG производится уже PostSript-интерпретатором[7]. К недостаткам утилиты можно отнести то, что в силу своей лицензии она не может распространяться со свободными дистрибутивами, а к достоинствам — отсутствие зависимостей.

Более комплексными решениями являются утилиты `sam2p` из одноимённого пакета и `bmeps`. Их так же можно найти на CTAN в директориях `graphics/sam2p` и `support/bmeps`, соответственно. `sam2p` является своеобразным комбайном, который поддерживает множество растровых графических форматов, в то время как `bmeps` фокусируется на PNG и JPEG. Обе эти программы позволяют получить вполне приличную eps-картинку для печати или для просмотра на экране. В обоих случаях необходимо поразбираться в ключах и настройках. С моей точки зрения `bmeps` является более удобным решением, производящим достаточно маленькие[8] по размеру eps-файлы, но и `sam2p` достаточно хорош.

Опять же на CTAN в директории `graphics/a2ping` можно взять довольно увесистый perl-скрипт `a2ping.pl`. Этот скрипт является своеобразной надстройкой над

[3]Portable Network Graphics — растровый графический формат использующий сжатие без потерь.

[4]Joint Picture Experts Group — самый популярный графический формат использующий сжатие с потерями.

[5]Portable Document Format — платформонезависимый формат электронных документов, созданный Adobe System.

[6]The Comprehensive TeX Archive Network. Центральный сайт `http://www.ctan.org`.

[7]Это стало возможным начиная с версии PostScript Level 2

[8]Это важно, так как мало какой растровый редактор может оптимально сохранить в EPS.

sam2p и Ghostscript, что позволяет ему более-менее автоматически конвертировать из растра в PostScript и обратно.

Обзор внешних программ закончен. Далее слово «пакет» будет относиться к пакетам LaTeX, а не пакетам GNU/Linux-дистрибутива.

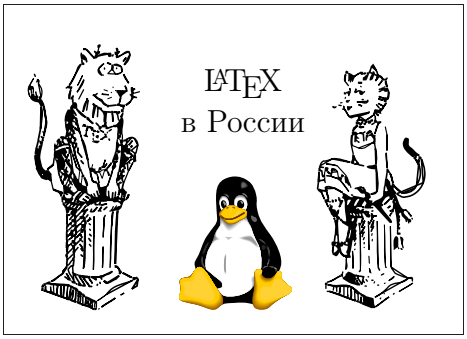
4.3. graphicx

Ответственным за создание «бокса» для размещения картинки является пакет **graphicx**[9], а точнее команда \includegraphics:

```
% Эмблемы \TeX{} и \METAFONT{}, созданные
%Дуайном Бибби, взяты со странички Д.Э.\,Кнута.

% Цветной пингвин взят из пакета \texttt{ps2pdf}
%от Rolf Nieprashk

\includegraphics[width=\textwidth]{title.1.eps}
```



В команде есть один обязательный параметр — вставляемая картинка. Необязательные параметры передаются с помощью пар «ключ»=«значение», разделяемых запятой. За подобный способ объявления параметров отвечает пакет **keyval**. Некоторые из поддерживаемых пакетом параметров перечислены ниже:

bb — позволяет исправить `BoundingBox` прямо в коде, не меняя eps. Значение представляет из себя четыре числа кодирующие положение левого нижнего угла и правого верхнего, например: `[bb=127 464 430 667]`. Вместо одного `bb`, можно воспользоваться четвёркой ключей: `[bbllx=127,bblly=464,bbrx=430,bbry=667]`, каждому из которых передаётся только одно значение.

Кроме перечисленных ключей для модификации `BoundingBox` можно использовать `viewport` — четыре числа значения описывают границы `BoundingBox`, где в качестве центра координат выбирается левый нижний угол уже существующего описания и `trim` — четыре числа значения описывают отступы от левой, нижней, правой и верхней границ.

clip — обрезает вставленную картинку по `BoundingBox`. Это необходимо сделать в случае изменения границ для «выкусывания» части картинки, иначе она будет «вылезать» за пределы выделенного её бокса. По умолчанию имеет значение `false`. Отсутствие значение у ключа `clip` при его упоминании эквивалентно значению `true`. Подобное поведение верно и для других логических переключателей.

[9] **graphicx** пришёл на смену пакету **graphics** — различия в последней букве. Команды из предыдущего пакета также можно использовать, но настоятельно не рекомендуется.

angle — поворачивает, картинку на указанный угол в градусах.

origin — определяет координаты центра вокруг которого вращается рисунок. Кроме непосредственно координат `origin` принимает и буквенные сокращения: `l`, `b`, `r` и `t` — соответствует центру вращения слева, снизу, справа и сверху. В этом случае выбирается середина указанной стороны. Возможны комбинации, задающие углы картинки: `lt`, `rt`, `rb` и `lb`. `c` — означает центр картинки.

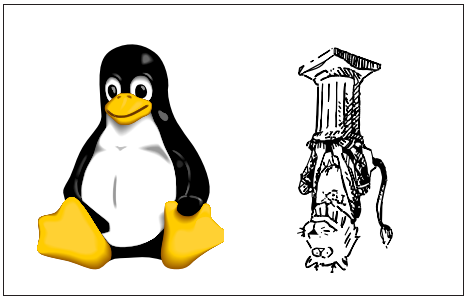
width — ширина вставляемой картинки.

height — высота вставляемой картинки.

scale — масштабный коэффициент.

keepaspectratio — логический переключатель. Модифицирует параметры высоты и ширины картинки в сторону уменьшения с целью сохранения естественных пропорций картинки.

```
\includegraphics[trim=110 0 105 100,clip,
            width=0.49\textwidth]{title.1.eps}
\hspace{0.5cm}
\includegraphics[viewport=0 0 100 200,clip,
              width=0.49\textwidth,
              height=3cm,keepaspectratio,
              angle=180,origin=c]{title.1.eps}
```



Аргументы \includegraphics интерпретируются слева направо. Для команд вращения и масштабирования порядок следования *имеет* значение.

Определение своих правил

Пакет **graphicx** предоставляет возможность перед вставкой картинки вызвать внешнюю программу для её обработки. Например, так можно добавить возможность включения в документ png-файлов:

```
\DeclareGraphicsRule{.png}{eps}{.bb}{`bmeps -p3 -c #1}
```

Первый параметр определяет расширение нового формата графики, для которого задаются правила. В представленном примере это `.png`. Второй параметр указывает тип графики. После преобразования это будет `eps` — `dvips` по умолчанию ничего другого и не знает. Третий параметр определяет расширение файла откуда считывать параметры `BoundingBox`. Файл должен содержать одну строчку вида:

```
%%BoundingBox: 0 0 848 979
```

До вставки для каждой png-картинки необходимо создать такой файл, например, следующим образом:

```
bmeps -b «картинка».png «картинка».bb
```

Последний параметр определяет команду, которую следует выполнить для преобразования картинки. Команда должна выдавать результат на стандартный вывод. `#1` соответствует имени обрабатываемого файла. Непосредственное выполнение команды происходит при трансляции dvi-файла.

Выполнение внешней команды является потенциально *опасной* процедурой, поэтому защита по умолчанию этого не позволяет. Для просмотра dvi-файла через `xdvi` следует использовать ключик `-allowshell`. В противном случае будет выдаваться запрос на исполнение команды каждый раз, когда встречается вставка по новым правилам. Для преобразования в PostScript в случае `dvips` также следует отключить защиту с помощью ключика `-R0`. Лучше всё-таки по возможности избегать вышеописанной процедуры и сразу готовить картинки в eps-формате.

Для трактования всех неизвестных драйверу расширений как eps следует применить команду:

```
\DeclareGraphicsRule{*}{eps}{*}{}
```

Это полезно в случае вставки картинок MetaPost, которые по умолчанию не имеют расширений. Если третий параметр равен «звёздочке», то это означает, что `BoundingBox` следует искать в том же файле, что и графику.

4.4. Плавающие объекты

Мало просто поместить картинку — её надо разместить красиво и по возможности она должна это делать самостоятельно. Просто \includegraphics для этого дела не очень подходит, так как размещение регулируется исключительно пользователем. Для этой цели в LaTeX имеется специальная сущность: *плавающий объект* (float[10]). Если для этого объекта нет места на текущей странице, то он переносится на следующую.

Для размещения картинок стандартные классы определяют плавающий объект как окружение `figure`:

```
\begin{figure}[ht]
  \centering%центрируем картинку
  \includegraphics{«картинка»}
  \caption{«подпись»}\label{fig:metka}
\end{figure}
```

В качестве необязательного параметра окружению `figure` можно передать допустимые способы размещения плавающего объекта:

h — разместить по возможности здесь же,

t — разместить в верхней части страницы,

[10] Пакет, который позволяет создавать новые типы плавающих объектов, так и называется **float**. Вместо этого пакета в качестве замены можно использовать **floatraw**, созданный Ольгой Лапко.

b — разместить в нижней части страницы,

p — разместить на отдельной странице, где нет ничего кроме плавающих объектов.

Приоритет для размещения определяется порядком следования букв. Если первой следует буква `h`, то в случае неудачи LaTeX размещает плавающий объект на *следующей* странице. Если же первыми следуют буквы `t` или `b`, то размещение организуется на текущей странице.

Для «красивого» размещения картинок LaTeX опирается на некоторые значения по умолчанию, которые не всегда для текущего случая могут быть оптимальными. Поэтому, если очень хочется разместить картинку, например, внизу, то пожелание можно усилить с помощью восклицательного знака: `[b!]`.

Управление плавающими объектами

Если плавающих объектов в документе немного, то всё будет хорошо без какого-либо вмешательства человека. Но если их много, то так или иначе надо будет управлять их размещением.

clearpage Если LaTeX не справляется с размещением картинок, то он переносит их на следующую страницу. В какой-то момент может накопиться целая «толпа» таких перенесённых картинок и возникнет необходимость в их «насильственном» выводе в каком-то определённом месте. Для этого существует команда **clearpage**. При вызове этой команды завершается текущая страница и выводятся все отложенные плавающие объекты и только потом продолжается обычный вывод текста. Единственная проблема этой команды в том, что текущая страница по ней обрывается. Чтобы избежать обрыва можно воспользоваться пакетом **afterpage**, точнее одноимённой командой из него:

```
\afterpage{\clearpage}
```

Команда \afterpage откладывает выполнение указанных в ней инструкций до конца текущей страницы.

suppressfloats Команда **suppressfloats** полностью подавляет размещение плавающих объектов на текущей странице. В качестве необязательного параметра ей можно передать `t` или `b` — в этом случае запрет распространяется только на размещение плавающих объектов вверху или внизу страницы соответственно.

placeins Пакет **placeins** не даёт «утекать» плавающим объектам за установленные пределы. Барьер устанавливает с помощью команды \FloatBarrier.

Это бывает полезно, когда хочется чтобы все картинки не выходили за пределы своего раздела. В этом случае следует переопределить нужную команду секционирования для установки перед ней барьеров. В случае команды секционирования раздела (`section`) достаточно передать опцию `[section]` пакету при загрузке:

```
\usepackage[section]{placeins}
```

endfloat Часто при подготовке статей требуют их размещения после текста на отдельных страницах предваряя эту галерею списком иллюстраций. Пакет **endfloat** именно это и делает. Достаточно его загрузить.

«Упаковка» картинок в один float

Для уменьшения «поголовья» плавающих объектов полезно размещать картинки группами. Например, чтобы разместить две картинки рядом можно применить команду **parbox** или окружение `minipage`:

```
\parbox[«позиционирование»]{«ширина»}{«текст»}

\begin{minipage}[«позиционирование»]{«ширина»}
 текст
\end{minipage}
```

В обоих случаях есть обязательный параметр ширина по которой формируется создаваемый бокс и необязательный «позиционирование» — расположение сформированного бокса относительно базовой линии по вертикали. Позиционирование может проводиться по центру (опция `[c]` — верно по умолчанию), по верхней линии (`[t]`) и по нижней линии бокса (`[b]`). Шаблон для двух рядом стоящих рисунков может иметь примерно следующий вид:

```
\begin{figure}[ht]\centering
  \parbox[b]{0.49\textwidth}{\centering
    \includegraphics{«рисунок-1»}
    \caption{«подпись-1»}\label{fig:metka-1}}
  \hfil\hfil%раздвигаем боксы по горизонтали
  \begin{minipage}[b]{0.49\textwidth}
    \centering
    \includegraphics{«рисунок-2»}
    \caption{«подпись-2»}\label{fig:metka-2}
  \end{minipage}
\end{figure}
```

Использование команды **parbox** или окружение `minipage` зависит исключительно от личных предпочтений. С помощью них можно организовать и более сложные конструкции.

Для целей автоматизации упаковки можно использовать и специализированные пакеты:

subfig — организует группы из множества картинок. Относительно современный пакет.

miniplot — делает то же что **subfig**, хоть и менее изощрённо.

figsize — специализируется на автоматическом вычислении размеров картинок для размещения их в указанных пределах.

dpfloat — определяет новый тип плавающего окружения, занимающего сразу две страницы. Двойные иллюстрации на развороте.

Картинки «в оборку»

Маленькие иллюстративные рисунки удобно делать в оборку с текстом, т. е. тест должен обтекать их. Такие картинки располагаются на внешней стороне страницы, т. е. слева для чётных и справа для нечётных страниц или в случае одностороннего режима печати.

Рис. 1. Подпись

Рис. 4.1. Картинка в оборку.

Традиционно описываются два пакета для создания подобных рисунков: **floatflt** и **wrapfig**. **floatflt** более автоматизирован для размещения картинок, но он так же чаще «ломается» при большом числе плавающих объектов. Возможны даже «потери» картинок. Упомянутые пакеты определяют окружения `floatingfigure` и `wrapfigure`, соответственно.

```
\begin{floatingfigure}[«размещение»]{«ширина»}
  ...
\end{floatingfigure}
```

Необязательный параметр «размещение» позволяет изменить алгоритм размещения картинки:

rflt — размещать справа,

lflt — размещать слева,

vflt — слева для чётных и справа для нечётных страниц (по умолчанию).

```
\begin{wrapfigure}[«число строк в оборке»]
                  {«размещение»}{«ширина»}
  ...
\end{wrapfigure}
```

В отличии от окружения `floatingfigure` `wrapfigure` требует определить правила размещения картинки. Доступные варианты: справа (`{r}`), слева (`{l}`), с внешней стороны страницы (`{i}`) и с внутренней стороны страницы (`{o}`). Если вместо строчных букв передать заглавные, то включается запрет на сдвиг по вертикали — картинка должна быть размещена начиная с той строки абзаца, в которой она была определена.

Необязательный параметр «число строк в оборке» позволяет указать число строк текста, которые должны быть сбоку от картинки. При этом выносная формула считается за три строки текста. Если параметр не определён, то число строк вычисляется автоматически, к сожалению не всегда оптимально.

Свою процедуру размещения картинки в оборку с текстом предлагает так же и пакет **nccfloats**[11]:

```
\sidefig(«ширина картинки»)(«ширина текста»)
{\includegraphics{«картинка»}}{«текст»}
```

В этом случае предлагается передавать команде \sidefig и саму картинку и текст, помещаемый сбоку. Параметр «ширина текста» можно опустить. Тогда текст занимает всё оставшееся пространство. Подробности можно посмотреть в документации `nccfloats.pdf`.

Подписи к рисункам

Для добавление подписи к рисунку используется команда **caption**, которую можно использовать только внутри плавающих объектов. В качестве обязательного параметра передаётся текст подписи. При выводе подпись центрируется, если она достаточно мала. В противном случае подпись оформляется в виде абзаца. Текст подписи не должен содержать команд разрыва строки. Все хрупкие команды внутри подписи должны быть защищены с помощью команды **protect**. **caption** можно передать также необязательный параметр, который должен представлять из себя краткую версию подписи, появляющуюся в автоматически создаваемых списках.

Оформление подписи жёстко привязано к стилю документа и изменить её без переопределения самой команды **caption** не просто. Для модификации параметров следует воспользоваться пакетами `caption` или `ccaption`. Оба упомянутые пакета позволяют «покрутить» множество ручек и снабжены исчерпывающей и очень объёмной документацией.

При включении русской опции **usepackage**[russian]{babel} перед подписью выводится слово `Рис.` за которым идёт автоматически вычисляемый порядковый номер картинки. В качестве разделителя между счётчиком и подписью по умолчанию используется двоеточие. Для замены двоеточия на точку в преамбуле достаточно набрать, например, следующее:

```
\usepackage{ccaption}
% заменяем для рисунков ':' после номера рисунка на '.'
\captiondelim{. } % после точки стоит пробел!
```

Кроме традиционного размещения подписи под картинками, подпись можно вынести, например на поля страницы. Не даром же стандартные классы имеют такие широкие поля. Пакет **mcaption** определяет окружение `margincap`:

[11] **nccfloats** входит в коллекцию **ncctools**, созданную А.И. Роженко

```
\begin{figure}[ht]
  \begin{margincap}{«Подпись»}
    \includegraphics{«картинка»}
  \end{margincap}
\end{figure}
```

В качестве обязательного параметра окружению передаётся подпись, а внутри определяется картинка.

4.5. Заключение

Странный получился рассказ. В статье, имеющей заголовок «Графика» не было ни слова сказано о том, как эту графику, собственно говоря, создавать. А ведь *есть* что сказать, но в данном случае информация о размещении и оформлении готовых картинок поважнее будет. Но и о том, как LaTeX умеет рисовать тоже будет рассказано, но чуть попозже.

Врезка: OpenSource шагает по стране. «Юзеры» наступают.

Посмотрите на картинку. Что это? Это TpX — простой векторный редактор, написанный под альтернативную операционную систему.

Чем же он интересен? Гххм, сложный вопрос. Ну да — его можно запустить в wine и от некоторых действий он даже не выпадает в осадок. Но это не Inkscape и отнюдь не xfig. Ещё один ничем не выдающийся велосипед, если бы не три «но».

Первое «но» в том, что этот «велосипед» под GPL (`http://sourceforge.net/projects/tpx/`).

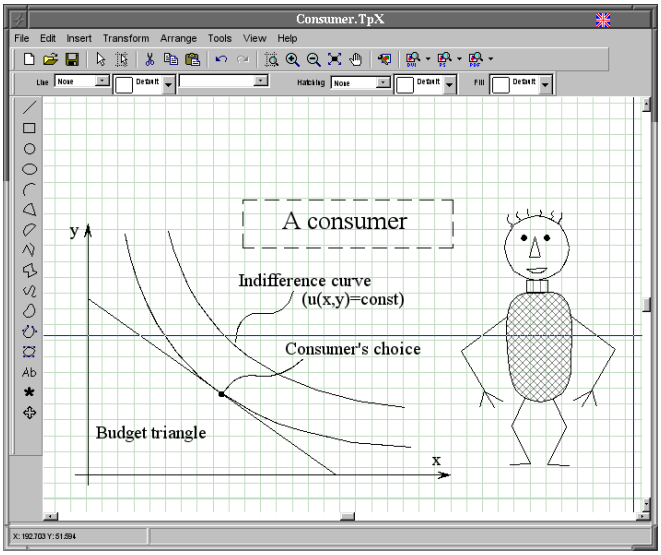


Рис. 4.2. TpX — простой векторный редактор.

Второе «но» заключается в специализации этого редактора — он предназначен для создании простых картинок с последующим внедрением в LaTeX. То есть это всё-таки специализированный велосипед.

«Но» третье и основное заключается в авторе. Это Александр Анатольевич Цыплаков (`http://www.nsu.ru/ef/tsy/`) — кандидат экономических наук и доцент Новосибирского государственного университета. Да, для разработки использовался Delphi, да информацию, что программа под GPL найти в коде не просто (её там нет) — отсутствие технической культуры, да диссертация написана в альтернативном текстовом редакторе — как это принято у экономистов. Но автор не технарь — он тот самый «юзер» о котором так любят рассуждать программисты. «Юзер» недовольный нехваткой инструментов настолько, что начал создавать их самостоятельно. Не просто создавать, а распространять результаты своего труда под свободной лицензией. И этой программой вполне можно пользоваться.

Правда, пока автор не освоит в совершенстве Lazarus, более-менее стабильной версии под Linux вряд-ли можно ожидать. Ну, естественно, если кто-нибудь ему не поможет, а то «юзеры» будут вынуждены взять власть в свои «очумелые» руки. ☺