ЧЕБЫШЕВСКИЙ СБОРНИК
Том 15 Выпуск 3 (2014)

УДК 511.6

О РЕАЛИЗАЦИИ *Logo* В LaTeX

А. Р. Есаян, А. В. Якушин (Тула)

Аннотация

В последнее время в публикациях для подготовки высококачественных рисунков все чаще используется свободно распространяемый графический пакет Ti*k*Z [3], ставший неотъемлемой частью научно-технической издательской системы TeX/LaTeX. В его библиотеке *turtle* реализован один из вариантов элегантного языка программирования высокого уровня Лого (*Logo*), разработанного в 1967 году Сеймуром Пайпертом (Seymour Papert) и Эдит Харель (Idit Harel) [1] и построенного на рекурсии.

Стоит отметить, что версия Лого, предоставляемая пакетом *turtle*, это лишь один из порядка 250 различных реализаций этого языка. Но, имея минимум средств, данная версия очень проста в освоении и позволяет программно, не выходя из LaTeX, без особых затруднений осуществлять несложное техническое рисование. О версии Лого, предоставляемой библиотекой *tirtle* пакета Ti*k*Z, и идет речь в данной статье. При этом описаны не только все средства этой версии, но и на иллюстративных примерах продемонстрировано их использование.

Заметим, что Лого – это не просто язык, а связка языка и интегрированной среды разработки, позволяющая придать простоту и наглядность процессу обучения программирования. В стандартном варианте Лого имеется исполнитель команд, который может перемещать по экрану точку (черепашку) вперед на заданное количество шагов и менять направление своего движения. При этом черепашка при перемещении может оставлять след на экране в виде отрезков линий, вычерчивая те или иные фигуры, или двигаться, не оставляя следа. Совокупность команд, понимаемая исполнителем “черепашка”, и составляет основу языка Лого, оказавшего в силу своей простоты революционное влияние на начальное обучение программированию. Понять Лого в состоянии дети младшего школьного и даже дошкольного возраста. Согласно выводам диссертационной работы Эдит Харель, разрабатывая дизайн программы, думая над тем, как лучше структурировать и представлять информацию, ученик глубже начинает понимать содержание задачи, лежащей в основании программы.

Отметим два немаловажных обстоятельства. В *tirtle*-реализации Лого смена направления движения черепашки может происходить под любым углом к текущему направлению ее движения, а для перемещения черепашки наряду со стандартными командами может быть использована и обычная структура цикла `\foreach`.

Ключевые слова: TEX, LATEX, Ti*k*Z, *tirtle*, Лого.

Библиография: 3 названия.

ON THE IMPLEMENTATION *Logo* IN LATEX

A. R. Esayan, A. V. Yakushin (Tula)

Abstract

Recently in publications for preparation of high-quality images are increasingly used freely distributed graphical Ti*k*Z package [3], which became an integral part of the scientific and technical publishing system TEX/LATEX. It's library *turtle* implemented one of the versions elegant programming language high level *Logo*, developed in 1967 by Seymour Papert and Idit Harel [1] and is based on recursion.

The version of the *Logo*, provided by the package *turtle*, this is just one of about 250 different implementations of the language. But, with a minimum of resources, this version is very easy to use and allows you to programmatically from the comfort of LATEX, easily supports simple technical drawing. About the version of the *Logo* provided by the library *tirtle* package Ti*k*Z, , and is discussed in this article. Here is described not only all means of this version, but also demonstrated their use on illustrative examples.

Note that the *Logo* this is not just a language, but a bunch of language and integrated development environment, allowing to give the simplicity and clarity to the process of teaching programming. In the standard version of the *Logo* has executor of the commands that can be moved point (*turtle*) around the screen forward on a specified number of steps and change the direction of its motion. This turtle when moving can leave a mark on the screen in the form of line segments describing these or other shapes, or move, without leaving a trace. A set of commands that are understood by the executor *turtle*, and is the basis of language *Logo*, which had in its simplicity a revolutionary impact on the start learning programming. To understand *Logo* in the state of children of primary school and even pre-school age. According to the conclusions of dissertational work Idit Harel, in developing the design of the program, thinking on how best to structure and present information, the student is much deeper begins to understand the content of the task that lies at the base of the program.

Note two important circumstances. In *tirtle*-implementation *Logo* change of direction *turtle* can occur at any angle to the current direction of its movement, and to move the *turtle* besides the standard commands can be used and the typical structure of a cycle `\foreach`.

Keywords: TEX, LATEX, Ti*k*Z, *tirtle*, *Logo*.

Bibliography: 3 titles.

1. Введение

Библиотека *turtle* пакета Ti*k*Z [3, с.739] открывается командой `\usetikzlibrary{turtle}`. Ее средства позволяют выводить изображения в стиле команд языка программирования Лого. Делается это следующим образом. На экране в определенной позиции находится виртуальная черепашка. Она всегда ориентирована по некоторому конкретному направлению. Исходная позиция черепашки – начало координат, а начальное направление движения – вверх. Специальными значениями-ключами опции *turtle* можно заставить перемещаться черепашку на любое расстояние по ее направлению и также изменять это направление. Кроме того, всегда можно без рисования переместить черепашку в начало координат или иную позицию, а также локально или глобально задать шаг перемещения. По умолчанию такой шаг равен 1cm. Черепашка всегда перемещается относительно своей последней точки и всегда по последнему направлению. Путь черепашки можно строить *tikz*-командой `\draw` по схеме:

```
\draw [turtle={...}] (a,b) [turtle={...}] (c,d)...[turtle={...}]   (1)
```

Запись (1) означает, что после вывода части изображения ключами первой опции *tirtle* черепашка без рисования перемещается в точку (a, b) с сохранениием направления движения. Далее ключами второй опции *turtle* выводится следующая часть изображения и опять без рисования черепашка перемещается в точку (c, d) и т. д. И еще один важный момент: структуры типа (1) часто удается оформить в виде циклов `\foreach`.

2. Ключи опции *turtle*

Опишем синтаксис и назначение отдельных ключей опции *turtle*:

- *home* – помещает черепашку в начало координат и задает ей направление “вверх”;
- $forward$ или fd – перемещает черепашку вперед по заданному направлению на расстояние di. Величина di берется из опции $distance = di$. Если эта опция отсутствует, то $di = 1cm$;
- $forward = di$ или $fd = di$ – перемещает черепашку вперед по заданному направлению на указанное расстояние di. Если di отрицательно, то фактически реализуется движение на величину $|di|$ в противоположном направлении;
- *right* или *rt* – задает направление поворота черепашки “направо” на угол 90°;
- $right = \alpha$ или $rt = \alpha$ – задает направление поворота черепашки “направо” на угол α°. Если величина угла α отрицательна, то фактически происходит поворот налево на $|\alpha|$ градусов;

- $left$ или lt – задает направление поворота черепашки “налево” на угол 90°;
- $left = \alpha$ или $lt = \alpha$ – задает направление поворота черепашки “налево” на угол α°. Если величина угла α отрицательна, то фактически происходит поворот направо на $|\alpha|$ градусов;
- $distance = di$ – меняет текущий шаг перемещения черепашки на di. Данный ключ можно использовать как опцию в начале окружения $tikzpicture$;
- $how/.style = \{out = \alpha, in = \beta, relative = lo\}$, где α и β – соответственно углы в градусах выхода и входа траектории черепашки на каждом шаге ее движения, $lo \in \{true, false\}$. То есть, α – это угол выхода из начальной точки и β – угол входа в конечную точку для каждого шагового фрагмента траектории черепашки. При опции $relative, relative = true$ и отсутствии опции $relative$ углы измеряются относительно прямой, соединяющей точки выхода и входа. При $relative = false$ углы измеряются в используемой системе координат абсолютно. Назначенный стиль действует до его изменения на все последующие шаги черепашки. Его можно использовать как в командах вывода траектории, так и в начале окружения $tikzpicture$. На практике часто применяются частные случаи рассматриваемого стиля вида $how/.style = \{b, relative\}$, где опция $relative$ необязательна, а параметр b может быть таким:
  - $bend\ left = \alpha$ – задает тот же стиль, что и последовательность $out = \alpha, in = 180^\circ - \alpha$ при изгибе влево;
  - $bend\ right = \alpha$ – выполняется как $bend\ left = \alpha$, но при изгибе вправо;
  - $bend\ left$ – выполняется как $bend\ left = \alpha$, где угол α берется из опции $bend\ angle = \alpha$;
  - $bend\ right$ – выполняется как $bend\ right = \alpha$, где угол α берется из опции $bend\ angle = \alpha$.

3. Стили $bend\ left$ и $bend\ right$

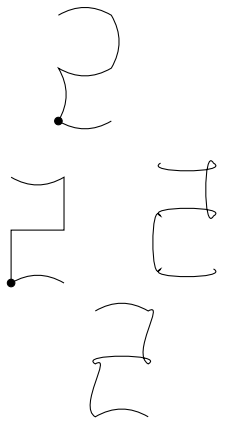
Здесь на примерах демонстрируется действие стилей $bend\ left$ и $bend\ right$. Результат выполнения кода показан после него.

```
%\usepackage{tikz} - в преамбулу
\usetikzlibrary{turtle}
\begin{tikzpicture}[turtle, distance=7mm,
      how/.style={bend right}]
  \draw [fill] (0,0) circle (0.5mm);
  \draw [turtle={home,fd,rt,fd,lt, fd,lt,fd,home,rt,fd}];
\end{tikzpicture}
```

```
\begin{tikzpicture}[turtle, distance=7mm]
  \draw [fill] (0,0) circle (0.5mm);
  \draw [turtle={home,fd,rt,fd,lt,fd,
    how/.style={bend right=-30},lt,fd,home,rt,fd}];
\end{tikzpicture}
```

```
\begin{tikzpicture}[turtle, distance=7mm]
  \draw [turtle={home}]
     (1,0) [turtle={how/.style={bend left=150},
        lt,fd,rt,fd,rt,fd,lt,fd,lt,fd}];
\end{tikzpicture}
```

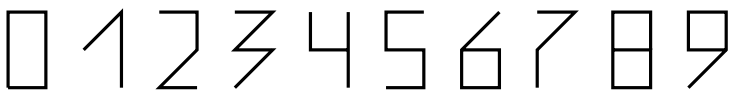
```
\begin{tikzpicture}[turtle, distance=7mm]
  \draw [turtle={home}]
     (1,0) [turtle={how/.style={bend left=150,
        relative=false}, lt,fd,rt,fd,rt,fd,lt,fd,lt,fd}];
\end{tikzpicture}
```



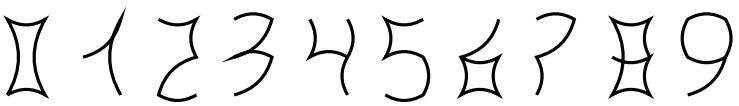
4. Индексы на почтовых отправлениях

Здесь цифры почтовых индексов выводятся отдельными командами `\draw` на предопределенные позиции. Результат представлен после кода на первом изображении. Два других изображения получены при выводе десятичных цифр стилями *bend right* и *bend left* соответственно.

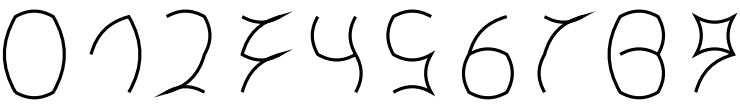
```
\usetikzlibrary{turtle}
\def\s{0.707}
\begin{tikzpicture}[very thick, turtle, distance=5mm]
  \draw [turtle={home,fd=1cm,rt,fd,rt,fd=1cm,rt,fd}];
  \draw [shift={(1,0.5)},turtle={home,rt=45,fd=\s,rt=135,fd=1cm}];
  \draw [shift={(2,1)},
     turtle={home,rt,fd,rt,fd,rt=45,fd=\s,lt=135,fd}];
  \draw [shift={(3,0)},
     turtle={home,rt=45,fd=\s,lt=135,fd,rt=135,fd=\s,lt=135,fd}];
  \draw [shift={(4.5,0.5)},
     turtle={home,fd,home,rt=180,fd,home,lt,fd,rt,fd}];
  \draw [shift={(5,0)},turtle={home,rt,fd,lt,fd,lt,fd,rt,fd,rt,fd}];
  \draw [shift={(6,0.5)},
     turtle={home,rt,fd,rt,fd,rt,fd,rt,fd,rt=45,fd=\s}];
  \draw [shift={(7,0)},turtle={home,fd,rt=45,fd=\s,lt=135,fd}];
  \draw [shift={(8,0.5)},
     turtle={home,rt,fd,rt,fd,rt,fd,rt,fd=1cm,rt,fd,rt,fd}];
  \draw [shift={(9,0)},
     turtle={home,rt=45,fd=\s,lt=45,fd,lt,fd,lt,fd,lt,fd}];
\end{tikzpicture}
```



% в опции *tikzpicture* добавить $how/.style = \{bend\ right\}$:



% в опции *tikzpicture* добавить $how/.style = \{bend\ left\}$:



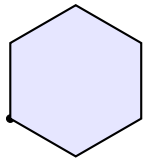
Представленный код в виде единой тропы можно оформить так:

```
\begin{tikzpicture}[very thick, turtle, distance=5mm]
  \draw [turtle={home,fd=1cm,rt,fd,rt,fd=1cm,rt,fd}]
      (1,0.5) [turtle={rt=135,fd=\s,rt=135,fd=1cm}]
      (2,1) [turtle={lt,fd,rt,fd,rt=45,fd=\s,lt=135,fd}]
      (3,0) [turtle={lt=45,fd=\s,lt=135,fd,rt=135,fd=\s,lt=135,fd}]
      (4.5,0) [turtle={rt,fd,lt,fd,rt,fd}] (4.5,0.5) [turtle={fd}]
      (5,0) [turtle={rt,fd,lt,fd,lt,fd,rt,fd,rt,fd}]
      (6,0.5) [turtle={fd,rt,fd,rt,fd,rt,fd,rt=45,fd=\s}]
      (7,0) [turtle={lt=45, fd,rt=45,fd=\s,lt=135,fd}]
      (8,0.5) [turtle={rt=180,fd,rt,fd,rt,fd,rt,fd=1cm,rt,fd,rt,fd}]
      (9,0) [turtle={lt=135,fd=\s,lt=45,fd,lt,fd,lt,fd,lt,fd}];
\end{tikzpicture}
```

5. Вывод правильных многоугольников

Начнем с вывода правильного шестиугольника. Если шесть раз подряд рисовать отрезок длиной a ($fd = a$ единиц) с разворотом направо на 60° ($rt = 60$), то мы и получим правильный шестиугольник. Отсюда и код:

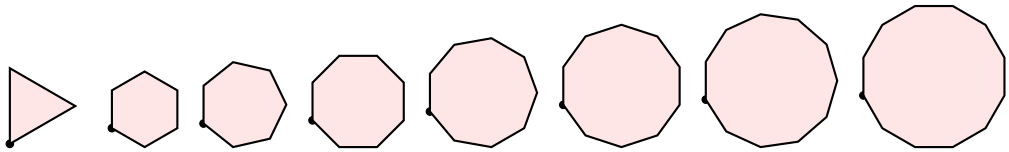
```
\usetikzlibrary {turtle}
\begin{tikzpicture}
  \draw [fill] (0,0) circle (0.5mm);
  \draw [fill=blue!10, thick, turtle=home]
     \foreach \t in {1,...,6}
        {[turtle={fd,rt=60}]};
\end{tikzpicture}
```



Здесь первая команда `\draw` используется лишь для указания начальной точки движения. По умолчанию $a = 1\text{cm}$.

Теперь напишем макрос с двумя аргументами для вывода произвольного выпуклого правильного n-угольника (`#1`$= n$) с длиной стороны a (`#2`$= a$). Поскольку внутренние углы такого многоугольника равны $(n-2)\cdot 180/n$, то при рисовании повороты следует производить на угол $180-(n-2)\cdot 180/n$ градусов. Отсюда и приведенный ниже код. По нему произведен вывод треугольника с длиной стороны в 1cm и в цикле – вывод шести-, семи-, ..., двенадцатиугольника со стороной 5mm.

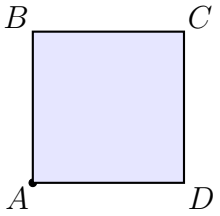
```
\usetikzlibrary {turtle, calc}
\newcommand{\repoly}[2]{
\def\alp{(180-(#1-2)*180/#1)}
\begin{tikzpicture}
  \draw [fill] (0,0) circle (0.5mm);
  \draw [fill=red!10, thick, turtle=home]
     \foreach \s in {1,...,#1}
        {[turtle={fd=#2,rt=\alp}]};
\end{tikzpicture}}

\noindent \repoly{3}{1}
\foreach \t in {6,...,12} {\repoly{\t}{5mm}}
```



6. Надписи у “пошаговых” точек в траектории черепашки

Здесь на примере вывода квадрата показано, как именовать точки в конце каждого шага траектории черепашки и выводить надписи около них.

```
\usetikzlibrary {turtle}
\begin{tikzpicture}
  \draw [fill] (0,0) circle (0.5mm);
  \draw [fill=blue!10, thick, turtle=home]
     \foreach \t/\s in {above left/B, above right/C,
                                below right/D, below left/A}
          {[turtle={fd=2cm,rt=90}] node [\t=-1mm] (\s) {$\s$}};
\end{tikzpicture}
```



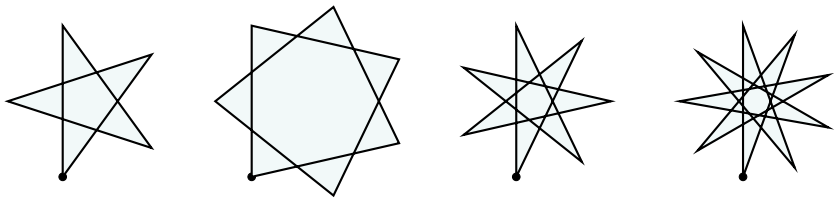
7. Вывод правильных звезд

Дадим определение понятия “звезда”. Разместим по кругу n точек на равном расстоянии друг от друга, то есть зададим правильный n-угольник. Далее,

каждую полученную точку соединим отрезком с m-ой ($1 \leqslant m \leqslant n/2$) по счету от нее точкой круга при движении по часовой стрелке. Полученную фигуру называют n/m-звездой или звёздчатой формой исходного выпуклого n-угольника. Вершинами n/m-звезды считаются исходные точки, а сторонами – проведенные отрезки. Точки пересечения сторон между собой вершинами звезды не считаются. Таким образом, n/m-звезда имеет n вершин и n сторон. У правильного выпуклого n-угольника может оказаться несколько звездных форм.

Далее по представленному коду выведены пятиугольная, две семиугольные и девятиугольная правильные звездные формы. Вся трудность их построения заключается в подсчете угла при смене направления движения черепашки. Для пятиугольной 5/2-звезды этот угол равен 144°, для семиугольной 7/2-звезды – (720/7)°, для семиугольной 7/3-звезды – (1080/7)° и для девятиугольной 9/4-звезды – 160°:

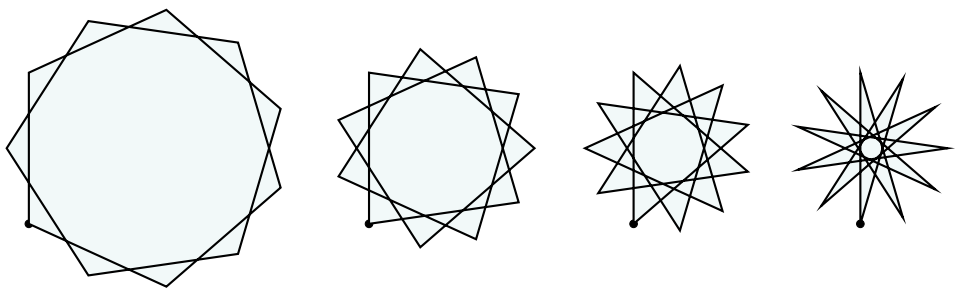
```
\usetikzlibrary {turtle}
\begin{tikzpicture}[turtle]
     \foreach \p/\q/\pos in
            {5/144/0, 7/{180*4/7}/2.5, 7/{180*6/7}/6, 9/160/9}
        {\draw [fill] (\pos,0) circle (0.5mm);
        \draw [fill=teal!5, thick] (\pos,0)
           \foreach \t in {1,...,\p}
             {[turtle={fd=2cm,rt=\q}]};}
\end{tikzpicture}
```



Если в предыдущем коде список первой команды `\foreach` заменить списком

$$11/180 \cdot 4/11/0, 11/180 \cdot 6/11/4.5, 11/180 \cdot 8/11/8, 11/180 \cdot 10/11/11,$$

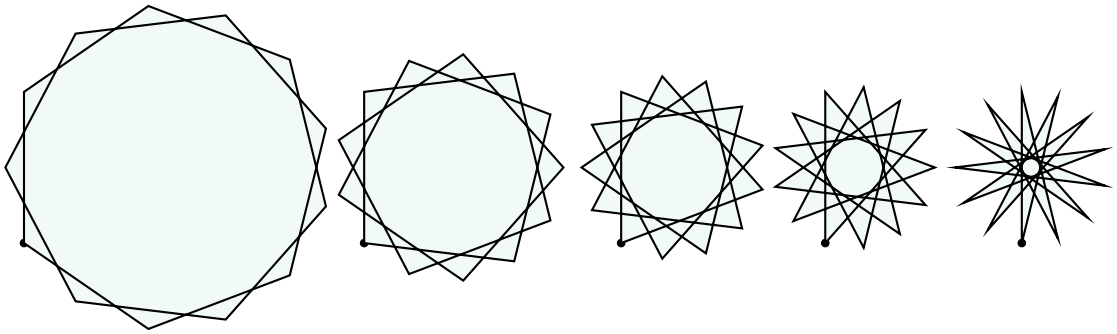
то для правильного выпуклого 11-угольника получим следующие четыре $11/m$-звезды при $m = 2, 3, 4$ и 5:



Если в исходном коде список первой команды `\foreach` заменить списком

$$13/180 \cdot 4/13/0, 13/180 \cdot 6/13/4.5, 13/180 \cdot 8/13/7.9,$$
$$13/180 \cdot 10/13/10.6, 13/180 \cdot 12/13/13.2$$

то получим следующие пять $13/m$-звезд при $m = 2, 3, 4, 5$ и 6:



8. Заключение

Из представленного материала вытекает, что средства библиотеки *tirtle* пакета Ti*k*Z легки в освоении и во многих случаях могут существенно упростить создание технических чертежей при работе в научной издательской системе TEX/LATEX.

СПИСОК ЦИТИРОВАННОЙ ЛИТЕРАТУРЫ

1. Пейперт С., Переворот в сознании. Дети, компьютеры и плодотворные идеи. М.: Педагогика, 1989. 224 с.

2. http://www.ctan.org/pkg/ - перечень пакетов расширений LATEX хранилища CTAN (Comprehensive TEX Archive Network)

3. Tantau T. TikZ and PGF Packages, manual for Version 3.0 / Till Tantau, Dec. 20, 2013. p. 1165, http://www.ctan.org/pkg/pgf

Тульский государственный педагогический университет им. Л. Н. Толстого
Поступило 4.02.2014