Paradigms: It's all in the game

Kees van der Laan

1 BLUe's Design VI

Hi folks. In 1989 Greene contributed 'Playing in TEX's mind' to TUG 89. Having fun with TEX has been on my mind all the time. Of late, I thought of tic-tac-toe as an exercise in programming dialogues in TEX, in a simple and elegant way, with hopefully some paradigms emerging, suited for educational purposes. The play is simple but nevertheless entails essential aspects in dialoging. Below I first strip the play to the essentials and from there build the user-friendly but otherwise modest code. At the end the TEXing peculiarities are summarized.

2 The play

Tic-tac-toe is played by two persons on a board with 3×3 fields. Each player marks a field in turn and the one who has first three marks in a row has won.

+	o	+
	o	+
o		+

3 Prototyping

The dialogue with TEX goes through the log file, also called transcript.

The board is represented by the defs `\1`, …`\9` rowwise. Each player owns a mark called `\markplayer` and `\markopponent`. A user is prompted to input a value—1, 2, ... or 9—which indicates the field where to put his mark. The program keeps track of the kind of mark, via a toggle. To end the game the user can input 0.

The above resulted in the following bare-to-the-bones implementation of tic-tac-toe.

```
\def\showboard{\immediate\write0{\1\2\3}
               \immediate\write0{\4\5\6}
               \immediate\write0{\7\8\9}}
\def\initialize{\def\1{-}\def\2{-}\def\3{-}
                \def\4{-}\def\5{-}\def\6{-}
                \def\7{-}\def\8{-}\def\9{-}}
\def\play{\initialize \loop\showboard
   \ifx\mark\markplayer
      \let\mark\markopponent\else
      \let\mark\markplayer\fi
   \immediate\write0{Supply index for \mark:}
   \read0to\index         \expandafter
   \xdef\csname\index\endcsname{\mark}
\ifnum\index>0 \repeat}%end \play
\def\markplayer{+}\def\markopponent{o}
\endlinechar-1 %TB20.18
\play \bye
```

Remarks. Because of the toggling of the `\mark` an `\xdef` was needed.

One could code that the player and opponent are both taking care of during each traversal of the loop. However, apart from that it makes the code longer it is also clumsy, and if careless one has to account for multiple exits of the loop. I assumed states in which the loop is traversed. The next steps are not for the faint of heart.

4 Real-life version

Given the above prototype implementation we can refine, add more bells-and-whistles. Basically these add-ons are in two directions

- improve the user interface
- let the program do more, such as deciding who has won.

4.1 Improving the user interface

Explain the conventions adopted, especially what the indices stand for.

Always nice is to allow for personalization, i.e. the system prompt asks for YOU, with your name. This entails that the players must be asked to identify themselves and that the toggling must be extended.

Another feature is that the program prompts the remaining indices to choose from.

And what about robustness? I decided not to implement robustness with respect to lowercase or uppercase y or n, for example. I also refrained from checking whether the supplied index is allowed.[1]

4.2 Let the program do more

The most important aspect is to add intelligence to the program to check for a winner.

The play restarts automatically.

4.3 The code

The board is represented by the defs `\1`, `\2`, … `\9`, the 3×3 board row-wise.

The structure of the program is similar to the prototype, with `\play` elaborated and the check `\checkforgameend` added.

```
\immediate\write0{Tic-tac-toe
         Aug 1995, cgl@rc.service.rug.nl}
\let\ea\expandafter
```

[1] Not that difficult, actually, because the set of allowed indices is maintained.

```
\newcount\k\newcount\value\newcount\checksum
\newif\ifsol
\newtoks\set %Index set
\def\del#1{\def\lop##1#1##2\pol{\set{##1##2}}
   \ea\lop\the\set\pol}
%
\def\showboard{\immediate\write0{}
\immediate\write0{\1\2\3}
\immediate\write0{\4\5\6}
\immediate\write0{\7\8\9}}
%Initialization
\def\initialize{\set{123456789}\solfalse
   \immediate\write0{New names of players?
       (default \player\space and \opponent)}
   \read0to\yorn
   \if y\yorn
     \immediate\write0{Name player}
     \read0to\player
     \immediate\write0{Name opponent}
     \read0to\opponent
   \fi\k0
   \loop\advance\k1
       \ea\def\csname\the\k\endcsname{-}
     \ifnum\k<10
   \repeat
   \immediate\write0{Empty board}
}%end initialization
%Test for solution
\def\sol#1#2#3{{\advance\count#1\count#2
  \advance\count#1\count#3
  \ifnum\count#1=\checksum \global\soltrue\fi}}
\def\checkforgameend{%
\sol123\ifsol\message{\who\space  won}\k0 \else
\sol456\ifsol\message{\who\space  won}\k0 \else
\sol789\ifsol\message{\who\space  won}\k0 \else
\sol147\ifsol\message{\who\space  won}\k0 \else
\sol258\ifsol\message{\who\space  won}\k0 \else
\sol369\ifsol\message{\who\space  won}\k0 \else
\sol159\ifsol\message{\who\space  won}\k0 \else
\sol357\ifsol\message{\who\space  won}\k0 \else
\fi\fi\fi\fi\fi\fi\fi\fi}
%Play
\def\play{\initialize\begingroup
\loop\showboard
   \ifx\who\player\value-1 \checksum-3
      \let\who\opponent
      \let\mark\markopponent
   \else\value1 \checksum3
      \let\who\player
      \let\mark\markplayer
   \fi
   \immediate\write0{\who, supply index
                    for \mark:}
   \immediate\write0{Choose from: \the\set.
         (0 terminates)}
   \read0t\ea o\csname\who\endcsname
   \k\csname\who\endcsname
   \ea\xdef\csname\the\k\endcsname{\mark}
   \count\k\value
   \checkforgameend
\ifnum\k>0 \ea\del\ea{\the\k}
\repeat\endgroup
\immediate\write0{Play another game?}
\read0to\newplayyorn
\if y\newplayyorn\ea\play\fi}%end Play
%Defaults
\def\player{Kees} \def\opponent{Ina}
\def\markplayer{+} \def\markopponent{o}
%
\immediate\write0{}
\immediate\write0{Board numbering}
\immediate\write0{123}
\immediate\write0{456}
\immediate\write0{789}
\play \bye
```

Remark. In order to get the personalized prompts `\Kees`, or `\Ina` the following coding was needed.

```
\read0t\ea o\csname\who\endcsname
```

The few lines that follow look unnecessary complex but are entailed by the above.

5 What more?

It is intriguing to ponder about adding even more intelligence. For example to let the game prompt for obligatory moves, or to let the program terminate when draw is inevitable, that is when there is no possible solution left. In order to achieve this I chose to

- maintain the set of possible solutions, instead of checking all possible solutions[2]
- update the set of solutions after each move, and look for a solution or a draw
- look for obligatory moves.

With respect to robustness the input can be checked for whether the index is allowed, casu quo a y(es) or n(o).

```
\immediate\write0{Tic-tac-toe
          Aug 1995, cgl@rc.service.rug.nl}
\let\ea\expandafter \let\nx\noexpand
\newcount\k\newcount\kk
\newcount\value\newcount\checksum
\newcount\feasible\newcount\prompt
%Solution lines
\newcount\hi\newcount\hii\newcount\hiii
\newcount\vi\newcount\vii\newcount\viii
\newcount\di\newcount\dii
\newif\ifsol
\newif\ifnotfound
%Index set and deletion from index set
\newtoks\set %Index set
\def\del#1{\def\lop##1#1##2\pol{\set{##1##2}}
   \ea\lop\the\set\pol}
\def\showboard{\immediate\write0{}
\immediate\write0{\1\2\3}
\immediate\write0{\4\5\6}
\immediate\write0{\7\8\9}}
%Initialization
\def\initialize{\set{123456789}\solfalse
   \hi0 \hii0 \hiii0 \vi0 \vii0\viii0
   \di0 \dii0 \feasible8 \prompt0
%Cell no associated with solution subsets
\ea\def\csname solset1\endcsname{%
   \ls\hi\ls\vi\ls\di}
\ea\def\csname solset2\endcsname{%
    \ls\hi\ls\vii}
\ea\def\csname solset3\endcsname{%
    \ls\hi\ls\viii\ls\dii}
\ea\def\csname solset4\endcsname{%
    \ls\hii\ls\vi}
\ea\def\csname solset5\endcsname{%
    \ls\hii\ls\vii\ls\di\ls\dii}
\ea\def\csname solset6\endcsname{%
    \ls\hii\ls\viii}
\ea\def\csname solset7\endcsname{%
    \ls\hiii\ls\vi\ls\dii}
\ea\def\csname solset8\endcsname{%
    \ls\hiii\ls\vii}
\ea\def\csname solset9\endcsname{%
    \ls\hiii\ls\viii\ls\di}
   \immediate\write0{New names of players?
```

[2]This entails that non-feasible candidates are eliminated from the set of candidate solutions.

```
    (default \player\space and \opponent)}
   \read0to\yorn
   \if y\yorn
     \immediate\write0{Name player}
     \read0to\player
     \immediate\write0{Name opponent}
     \read0to\opponent
   \fi\k0
   \loop\advance\k1
       \ea\def\csname\the\k\endcsname{-}
     \ifnum\k<10
   \repeat
   \immediate\write0{Empty board}
}%end initialization
%Test for solution
\def\ls#1{%#1 is a solution counter
  \ifnum#1=0 \advance#1\value
  \else\ifnum\sign#1=\value \advance#1\value
       \else\advance\feasible-1
          \delete#1 %from solution sets
          \ifnum\feasible=0 \showboard
              \message{***Draw***}\k0
          \fi
       \fi
  \fi
  %Check for solution
  \ifnum#1=\checksum \showboard
       \message{***\who*** won}\k0
  \fi}
\def\sign#1{\ifnum#1>0 1\else -1\fi}
\def\solsetk{\csname solset\the\k\endcsname}
\def\delete#1{%#1 is a solution counter to be
              %deleted from all solution sets
{\kk0
\def\ls##1{\ifx#1##1\else\nx\ls\nx##1\fi}
\loop\advance\kk1
\ifnum\kk<10
\ea\xdef\csname solset\the\kk\endcsname
    {\csname solset\the\kk\endcsname}
\repeat}}%end \delete
\def\fifo#1{\ifx#1\ofif\ofif\fi
   \process#1\fifo}
\def\ofif#1\fifo{\fi}
\def\lstry#1{%#1 is a  counter denoting
             %a solution
   \ifnum#1=0
   \else\ifnum\sign#1=\value
            \advance#1\value
        %Check for solution
        \ifnum#1=\checksum \global\prompt\k
        \fi\fi
   \fi}
\def\process#1{\ifnum\prompt=0
  {\k=#1 \let\ls\lstry \solsetk}\fi}
\def\readprocess#1{\if#1\csname\who\endcsname
   \notfoundfalse\fi}
\def\readindex{{\let\process\readprocess
   \loop\read0t\ea o\csname\who\endcsname
   %\who value in \set?
   \notfoundtrue\ea\fifo\the\set0\ofif
   \ifnotfound
   \immediate\write0{Please supply index
                        from \the \set}
   \repeat
   \global\k\csname\who\endcsname}}
%Play
\def\play{\initialize\begingroup
\loop\showboard%\show\ the\solset
   \ifx\who\player\value-1 \checksum-3
      \let\who\opponent
      \let\mark\markopponent
   \else\value1 \checksum3
      \let\who\player
      \let\mark\markplayer
   \fi
  %Is there a winning move?
  %(Prefails obligatory move)
  {\prompt0 \ea\fifo\the\set\ofif
   \ifnum\prompt>0 \global\prompt0 \fi}
  %The idea is that the player must see it
  %him/herself. Of course the program knows
  %about the winning move.
  %Obligatory move? (criterion 0<\prompt (<10)
  \ifnum\prompt>0 \immediate\write0{Sorry \who,
          obligatory move \the\prompt}\k\prompt
  \else
      \immediate\write0{\who, supply index
                    for \mark:}
      \immediate\write0{Choose from: \the\set.
         (0 terminates)}
      \readindex
  \fi
  %put mark on board
  \ea\xdef\csname\the\k\endcsname{\mark}
  %update solutions associated with \k
  \solsetk
%k=0 is stopping criterion
\ifnum\k>0 \ea\del\ea{\the\k}%
 %Look ahead for obligatory move:
 %    Can \who gain in next turn?
 \prompt0 \ea\fifo\the\set\ofif
\repeat\endgroup
\immediate\write0{Play another game?}
\read0to\newplayyorn
\if y\newplayyorn\ea\play\fi}%end Play
%Defaults
\def\player{Kees} \def\opponent{Ina}
\def\markplayer{+} \def\markopponent{o}
%And off we go
\immediate\write0{}
\immediate\write0{Board numbering}
\immediate\write0{123}
\immediate\write0{456}
\immediate\write0{789}
\immediate\write0{}
\play \bye
```

6 Is this all?

Especially the code in the last section has become complex. Should we make it more complex? For the moment I stopped. However, it is tempting to increase the order to 4, and to extend the play to 3 dimensions. My case rest.

7 Paradigms

The following functionalites have been encountered and a way how to code these in TEX have been worked out.

- loop traversal in various states
- global Boolean, `\global\soltrue`
- maintaining an index set, deleting an element from a set
- test for an empty set
- high-level parameterization, for example a personalized prompt within a state-dependent loop
- `\read` with a *reference* to a prompt through `\read0 t\ea o\csname\who\endcsname`
- check on what is read is allowed.

Starting from the bare-to-the-bones prototype it is quite something to arrive at a solution, which carries some intelligence and robustness .

Have fun, and all the best.