Tic Tac Toe
by Qianji Zheng and J. Nelson Rushton, Texas Tech University

This is an LED program that defines a simple tic tac toe game to run in the easel game engine.

The game begins with an empty grid and it being x's turn to move. When an empty cell is clicked, the player whose turn it occupies that cell and it becomes the other player's turn, until the game is over. When the game is over a message is displayed giving the result of the game. The player can press the 'reset' button at any time to restart the game.

```
=========================================================================
DATA MODEL
=========================================================================
```

A *player* is either `x or `o. In this program, the variable p will range over players.

A *cell* is an integer in {1..9}. In this program, the variable c will range over cells.

Cells represent squares on the tic tac toe board as pictured below:

```
1|2|3
-----
4|5|6
-----
7|8|9
```

A *move* is a pair (p,c) where p is a player and c is a cell. The move (p,c) represents a move by player p in cell c. In this program, the variable m will range over moves.

A *state* is a set of moves, thought of as the set of moves made far in the game. In this program, the variable S will range over states.

Global variables:
- *CURRENT_STATE*: same type as user-defined *initialState*
- *CLICKED*: true/false
- *CLICK_X*, *CLICK_Y*: integer

```
=========================================================================
GAME RULES
=========================================================================
```

```
This section defines the rules of tic-tac-toe in LED.

Since the state of the game is the set of moves that have been made, the
   beginning state *initialState* is the empty set.
/$
initialState := {}
$/

Player p *occupies* cell c if the move (p,c) is a member of CURRENT_STATE.
   Cell c is *occupied* if it is occupied by `x or by `o.
/$
occupies(p,c) iff (p,c) in CURRENT_STATE

occupied(c) iff occupies(`x,c) or occupies(`o,c)
$/

A *row* is a set of cells that form three in a row either horizontally,
   vertically, or diagonally.
/$
rows := hRows U vRows U diagonals where
    hRows = {{1,2,3},{4,5,6},{7,8,9}} &
    vRows = {{1,4,7},{2,5,8},{3,6,9}} &
    diagonals = {{1,5,9},{3,5,7}}
$/

*threeInRow(p)* if player p occupies all of the cells in some row.
/$
threeInRow(p) iff some R in rows. all c in R. occupies(p,c)
$/

*boardFull* if and only if all cells are occupied.
/$
boardFull := |CURRENT_STATE| = 9
$/

*gameOver* if either the board is full, or one of the players three in a row
   .
/$
gameOver := boardFull or threeInRow(`x) or threeInRow(`o)
$/

*playerToMove* is `x if an even number of moves have been made, and `o
   otherwise.
/$
playerToMove :=
    `x if even(|CURRENT_STATE|);
```

```
    ‘o otherwise

even(n) iff n mod 2 = 0
$/

*legalToMoveIn(c)* means that it is legal for the player whose turn it is to
    move in cell c in the current state of the game -- that is, if the game
    is not over and the cell is not occupied.
/$
legalToMoveIn(c) iff ~occupied(c) & ~gameOver
$/

=========================================================================
VIDEO OUTPUT
=========================================================================

The default color used in this game is BLACK
/$
BLACK := color(0, 0, 0)
WHITE := color(255, 255, 255)
BLUE := color(0, 0, 255)
GREEN := color(0, 255, 0)
RED := color(255, 0, 0)
$/

-------------------------------------------------------------------------

This section defines the *display* function, specifying the images to
   display on the screen in each game state. The *grid* consists is a set
   of four line segments that make up the tic tac toe playing area.
/$
gridDisplay := {L1,L2,L3,L4} where
    L1 = segment(point(200,700),point(200,400),BLACK) &
    L2 = segment(point(300,700),point(300,400),BLACK) &
    L3 = segment(point(100,600),point(400,600),BLACK) &
    L4 = segment(point(100,500),point(400,500),BLACK)
$/

The default font size for displayed text in this program is 36.
/$
fontSize := 36
$/

*centerX(c)* and *centerY(c)* are the x and y coordinates of the center of
   cell c, respectively.
/$
```

```
centerX(c) := 150 + 100 * ((c-1) mod 3)
centerY(c) := 650 - 100 * (floor((c-1)/3))
$/

*cellDisplay(c)* is a display of a text character "x" or an "o" in cell c,
   or the empty display, respectively in case cell c is occupied by ‘x,
   occupied by ‘o, or not occupied in the current game state.
/$
xImage(c) := text("x", point(centerX(c), centerY(c)), fontSize, BLUE)

oImage(c) := text("o", point(centerX(c), centerY(c)), fontSize, GREEN)

cellDisplay(c) :=
    {xImage(c)} if (‘x, c) in CURRENT_STATE;
    {oImage(c)} if (‘o, c) in CURRENT_STATE;
    {} otherwise
$/

*cellDisplays* is the set of all images of x’s and o’s on the board in the
   current state.
/$
gameBoard := {1..9}

cellDisplays := U[c in gameBoard] cellDisplay(c)
$/


----------------

If the game is not over, *currentPlayerDisplay* is a text in the upper left
   hand region of the game window indicating the player to move, either "
   play x’s turn" or "play o’s turn",
/$
currentPlayerDisplay :=
    {text("x’s turn",point(100,750),fontSize,BLACK)} if playerToMove = ‘x;
    {text("o’s turn",point(100,750),fontSize,BLACK)} otherwise
$/

The *restart button* consists of a rectangle around a "restart" text,
   displayed in the upper right region of the screen. Formally, it is a set
    of four line segments and a text.
/$
restartLeft := 350
restartRight := 550

restartBottom := 725
restartTop := 775
```

```
restartBottomLeftPoint := point(restartLeft, restartBottom)
restartBottomRightPoint := point(restartRight, restartBottom)

restartTopLeftPoint := point(restartLeft, restartTop)
restartTopRightPoint := point(restartRight, restartTop)

mid(a, b) := (a + b) / 2

restartMidX := mid(restartLeft, restartRight)
restartMidY := mid(restartBottom, restartTop)

restartButton := {A1,A2,A3,A4,txt} where
    A1 = segment(restartBottomLeftPoint,restartBottomRightPoint,BLACK) &
    A2 = segment(restartTopLeftPoint,restartTopRightPoint,BLACK) &
    A3 = segment(restartBottomLeftPoint,restartTopLeftPoint,BLACK) &
    A4 = segment(restartBottomRightPoint,restartTopRightPoint,BLACK) &
    txt = text("restart",point(restartMidX,restartMidY),fontSize,BLACK)
$/

If the game is over, *gameResultDisplay* is a display in the upper left
   region of the screen telling the outcome of the game, either "x won", "o
    won", or "cat got it"
/$
gameResultDisplay :=
    {text("x won",point(200,750),fontSize,BLUE)} if threeInRow(`x);
    {text("o won",point(200,750),fontSize,GREEN)} if threeInRow(`o);
    {text("cat got it",point(200,750),fontSize,RED)} otherwise
$/

The *display* is the screen display for the current game state. The grid,
   cell displays, and restart button are always displayed. Additionally,
   the display includes the game results if the game is over, and the
   player to move if the game is not over.
/$
images :=
    gameOverDisplay if gameOver;
    inPlayDisplay otherwise where
        alwaysDisplay = gridDisplay U cellDisplays U restartButton &
        inPlayDisplay = alwaysDisplay U currentPlayerDisplay &
        gameOverDisplay = alwaysDisplay U gameResultDisplay
$/

=====================================================================
MOUSE INPUT
=====================================================================
```

```
This section defines *update*, which specifies the program's response to
   mouse input. The variable *pt* will vary over points.

*xMin(c)*, *xMax(c)*, *yMin(c)*, and *yMax(c)* denote the graphical
   boundaries of cell c, in the obvious manner.
/$
xMin(c) := 100+100*((c-1) mod 3)
xMax(c) := 200+100*((c-1) mod 3)
yMin(c) := 600-100*(floor((c-1)/3))
yMax(c) := 700-100*(floor((c-1)/3))
$/

cellClicked(c) means that cell c has been clicked.
/$
cellClicked(c) iff
    CLICKED &
    CLICK_X > xMin(c) & CLICK_X < xMax(c) &
    CLICK_Y > yMin(c) & CLICK_Y < yMax(c)
$/

*restartClicked* means that the most recent mouse click is inside the region
    of the play again button.
/$
restartClicked :=
    CLICKED &
    CLICK_X > restartLeft & CLICK_X < restartRight &
    CLICK_Y > restartBottom & CLICK_Y < restartTop
$/


----------------

*moveMade(c)* means that cell c has been clicked and the player to move may
   legally move there.
/$
moveMadeIn(c) iff cellClicked(c) & legalToMoveIn(c)

movesMade := {(playerToMove,c) | c in gameBoard & moveMadeIn(c)}

newState :=
    initialState if restartClicked;
    CURRENT_STATE U movesMade otherwise
$/
```