Tic Tac Toe

Nelson Rushton, Qianji Zheng, and Vu Phan

November 6, 2016

Abstract

This program plays tic tac toe. 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.

1 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 $\{1..9\}$.

$$Cells \ := \ \{1..9\}$$

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.

The program references the constant symbols *currentState*, *clicked*, *clickX*, and *clickY*, which are of types *state*, *Bool*, *Int*, and *Int*, respectively.

2 Game Rules

This section defines the rules of tic-tac-toe. The top level definitions of this section are as follows:

- $initialState$ is the initial state of the game.
- $legalMove : move \rightarrow Bool$ — $legalMove(m)$ means that move m is legal in the current state.
- $outcome : move \rightarrow state$ — If m is a legal move in the current state, then $outcome(m)$ is the state that will result from making that move.
- $threeInRow : player \rightarrow Bool$ — $threeInRow(p)$ means player p has three in a row.
- $boardFull$ is a flag indicating that the board is full.

Since the state of the game is the set of moves that have been made, the initial state is simply the empty set.

$$initialState \;:=\; \emptyset$$

Player p *occupies* cell c if the move (p, c) is a member of $currentState$.

$$occupies(p, c) \;\equiv\; (p, c) \in currentState$$

Cell c is *occupied* if it is occupied by ‘x or by ‘o.

$$occupied(c) \;\equiv\; occupies(`x, c) \;\vee\; occupies(`o, c)$$

A *row* is a set of three cells that form three in a row either horizontally, vertically, or diagonally. We will write *rows* for the set of all rows.

$$\begin{aligned} rows &:= hRows \cup vRows \cup diagonals \;\; \text{where} \\ hRows &= \{\{1,2,3\},\{4,5,6\},\{7,8,9\}\}, \\ vRows &= \{\{1,4,7\},\{2,5,8\},\{3,6,9\}\}, \\ diagonals &= \{\{1,5,9\},\{3,5,7\}\} \end{aligned}$$

Player p *fills* row r if he or she occupies every cell in r.

$$fills(p, r) \;\equiv\; \forall c \in r.\; occupies(p, c)$$

We write $threeInRow(p)$ if player p fills a row.

$$threeInRow(p) \;\equiv\; \exists r \in rows.\; fills(p, r)$$

The board is full if all 9 cells are occupied.

$$boardFull \;\equiv\; |currentState| = 9$$

The game is over if either the board is full, ‘x has three in a row, or ‘o has three in a row.

$$gameOver \;\equiv\; boardFull \;\vee\; threeInRow(‘x) \;\vee\; threeInRow(‘o)$$

$playerToMove$ is ‘x if an even number of moves have been made, and ‘o otherwise,

$$playerToMove \;:=\; \begin{cases} ‘x & \text{if } even(|currentState|) \\ ‘o & \text{otherwise} \end{cases}$$

where, of course, an integer is *even* iff it is divisible by 2:

$$even(n) \;\equiv\; n \bmod 2 = 0$$

It is legal to move in cell c if the game is not over and c is not occupied.

$$legalToMoveIn(c) \;\equiv\; \neg gameOver,\; \neg occupied(c)$$

$moveMade$ means that a move is being made in some cell. The predicate $moveMadeIn : Cell \to Bool$ is defined in Section 3 (Mouse Input).

$$moveMade \;\equiv\; \exists c \in Cells.\; moveMadeIn(c)$$

If $moveMade$ holds, then $move$ is a singleton set whose only member is the move that is being made.

$$move \;:=\; \{(playerToMove, c) \mid c \in Cells,\; moveMadeIn(c)\}$$

The last two definitions give preconditions and effects of actions in the game. The actions ‘$restart$ and ‘nil are always possible. A move is possible if it satisfies the relation $legalMove$.

If the makes a move, it is added to the current state. If he restarts the game, it returns to its initial state. Otherwise, the state does not change.

$$nextState \;:=\; \begin{cases} currentState \cup move & \text{if } moveMade \\ initialState & \text{if } restart \\ currentState & \text{otherwise} \end{cases}$$

3 Video Output

This section defines the video output to the screen as a function of state.