SOEN 6011- Software Engineering Processes
(Summer 2016)
Project Team: SmartTech (Group 10)

Assignment 5- State Machine Diagram
On
"Tic-Tac-Toe"

Submitted by:
Vijay Shah (27735146)
Amandeep Sharma (27260164)
Bhawna Sharma (27568789)
Amritpal Singh (27684878)
Deepinder Singh (40004787)
Hardilpreet Singh (27822200)
Jatinderpal Singh (27727267)
Manvir Singh (27680120)

Submitted to:
Prof. Nicolangelo Piccirilli

Submission Date: June 3, 2016

Table of Contents

Table of Figures

1. Purpose

A UML State-machine diagram is a kind of behavior diagram depicting the behavior of a part of a designed system through finite state transitions. In other words, a state-machine diagram describes the various states that an object may be in and the transitions between those states. Hence, a state machine diagram plays a really important part of the documentation as it provides us with the complete information about the behavior of the system in terms of how the system is responding to the environment.

2. State machine diagram for heuristic

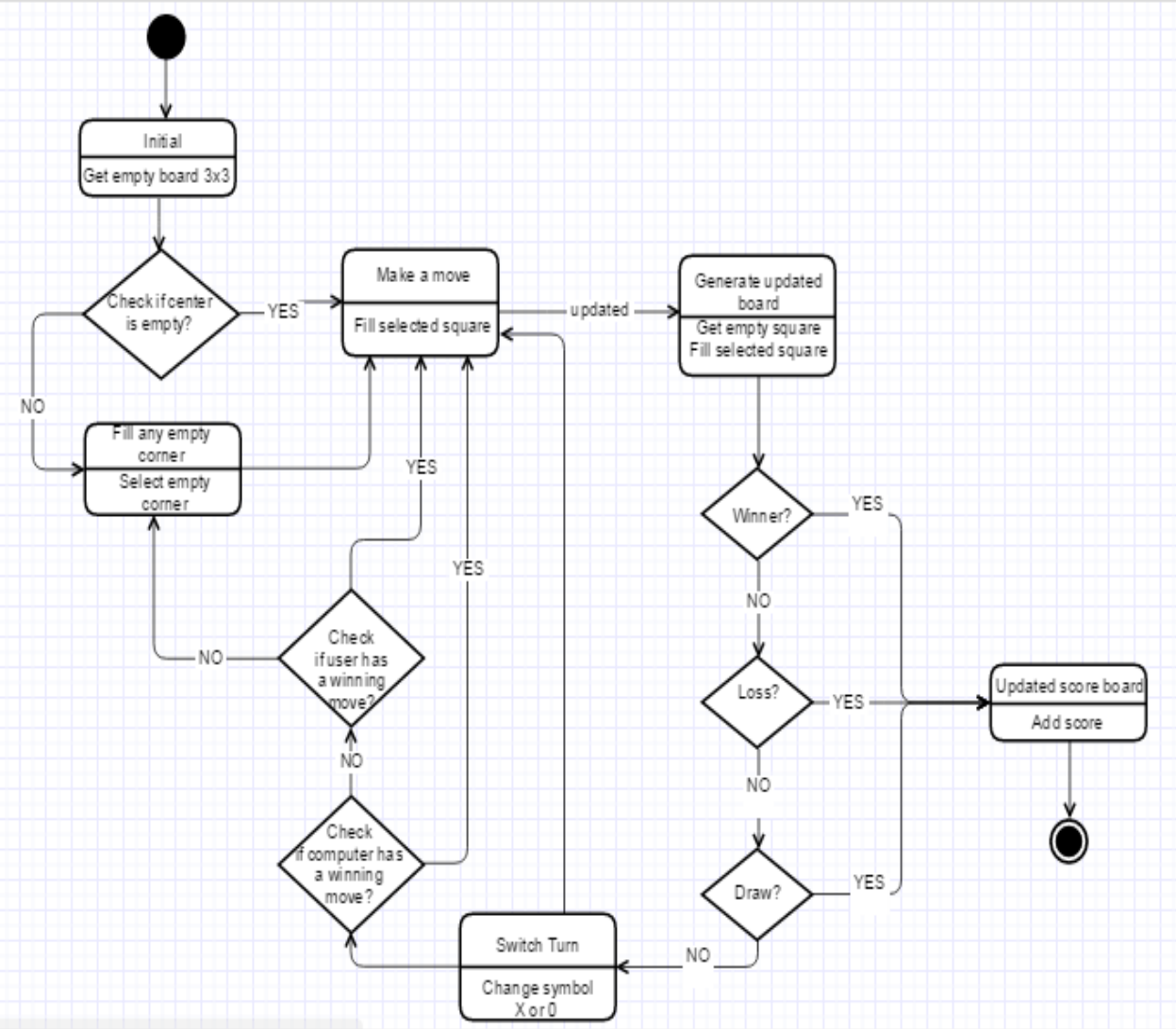

Figure 1: State Machine Diagram for Heuristic in project "Tic-Tac-Toe"

2.1 State machine diagram description

Above is our State machine diagram depicting our heuristic in Tic-Tac-Toe. It can be stated that once the game begins we have a 3x3 board (9 squares) which are empty and which will be filled by the end of the game resulting in either a win, lose or a draw situation. The computer in the advance version does not move random and plays according to the situation.

- As mentioned in our state machine diagram the computer would first check for the board to have an empty center and if the computer finds that the board is empty it would fill the center square by making the move because first move at center cell has the maximum chances for winning.
- In other case, if the center is not empty, the computer would then search for any corners that are not filled and would directly fill in one of the squares. (placing move at corner cells has the second maximum probability)
- Once the computer has made its move, the board is updated with the moves done by the player and computer alternatively and the system would check if the computer has won, lost or a draw situation and then update the score.
- If it's neither of those three (win, lose, draw) situation and we still have moves to be played on the board, then the computer would check if the board has place which once filled by the computer would result in the computer to win. If there is such move, then the computer would fill the square and the board would be updated.
- If the computer doesn't have any winning move, the next thing computer would do is to check if there is any user move that could make the result in the player to win the game, if such move is not found, the computer would then search for any corners that are not filled and would fill in one of the squares and then again system would check for the game status and loop goes forward. If user has any winning move, then the computer occupies that space and fills the square to block the user winning move and result in updating the board and finishing the game in a draw.

This describes completely the heuristics that the computer would consider in our Tic-Tac-Toe project.

3. References

[1] R. C. Martin . UML for Java Programmers 2003.

[2] H. v. Vliet . *Software Engineering : Principles and Practice 2000.*

[3] D. Smith . *Java for the World Wide Web 1998.*

[4] B. J. MacDonald . *Programming the Finite Element Method in Java and Android 2013.*

[5] Kurniawan, Budi.,Perry, Daniel.,. (). *Android*.

[6] Ableson, W. Frank., Sen, Robi.,King, Chris,. *Android in Action 2011*.