

**A Mini-Project Report**

ON

**IMPLEMENTATION OF TIC TAC TOE AI USING ZOBRIST HASHING**

BY

**MEDHA SHAMSUNDAR H.P. - 1PI13IS057**

**SWETHA B. - 1PI13IS114**

Under the guidance of

**Dr. SRINIVASA MURTHY**

**Department of CS**

**JANUARY 2015 – MAY 2**

**DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING**

**PES INSTITUTE OF TECHNOLOGY**

100 FEET RING ROAD, BANASHANKARI III STAGE  
 BANGALORE-560085



This is to certify that the Special Topic entitled Implementation of TicTacToe AI usign Zobrist Hashing is presented by **Medha Shamsundar H P, USN: 1PI13IS057** in partial fulfillment for the award of degree of Bachelor of Engineering in Information Science of the Visvesvaraya Technological University, Belgaum during the year 2014-2015. It is certified that all corrections / suggestions indicated for Internal Assessment have been incorporated in the report. The Special Topic has been approved as it satisfies the academic requirements in respect of Special Topic prescribed for the Bachelor of Engineering Degree.

Signature of the Guide Signature of the HOD

Dr. Srinivasa Murthy Dr. Shylaja S.S

**Acknowledgement**

The completion of any inter-disciplinary project depends upon cooperation, co-ordination and combined efforts of several sources of knowledge.

We are grateful to Dr.Srinivasa Murthy, Professor, CS, PESIT for his willingness to give us valuable advice and direction whenever we approached him with a problem. We are thankful to him for providing guidance for this project.

We are also thankful to and all the staff members, parents and friends for their cooperation and motivation in completion of our project.

ABSTRACT

Our project aims at implementing the game of Tic Tac Toe using a hashing technique called **Zobrist hashing.** We implemented it using Java and NetBeans (for UI).

Zobrist hashing is a hash function construction used in implementing computer board games like chess, go, etc. to implement transposition tables - a special kind of hash table that is indexed by a board position and used to avoid analyzing the same position more than once.

We used Java's jFrame class to create UI for our game. We implemented a two player game, player vs computer. We used AI techniques (Negamax algorithm), to make computer "think" for best position for its turn. We used Zobrist hashing to map all board positions with pawn (not empty position) to a random value, XOR all of them to get unique key and store that particular game state.

**INTRODUCTION**

**DESIGN CHOICES:**

Our design choice was Java from the beginning, as it is very code friendly and UI friendly. Since, we are implementing Tic Tac Toe, for a layman, just understanding the code is not enough. So, we used Java's jFrame to create UI and make the actual Tic Tac Toe board (using jButtons) along with few options, like, restart and reset.

The most difficult part of our project was the implementation of what we understood by Zobrist Hashing, transposition tables, Negamax algorithm, etc. We used Netbeans with all the options of jFrame provided to design and implement the UI.

**BACKGROUND:**

Tic Tac Toe is a simple two player game, and it is played by almost everyone, to kill their leisure time. It has 3X3 space. Two player, take their turns alternatively.

**Instructions for this game:**

1. X moves first.

2. A piece may be placed on any empty space.

3. A player wins by being the first to connect a line of friendly pieces from

one side or corner of the board to the other.

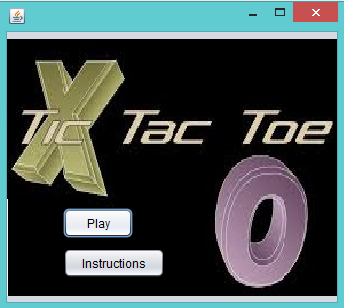
4. The game ends when either one player wins or it is no longer possible for a

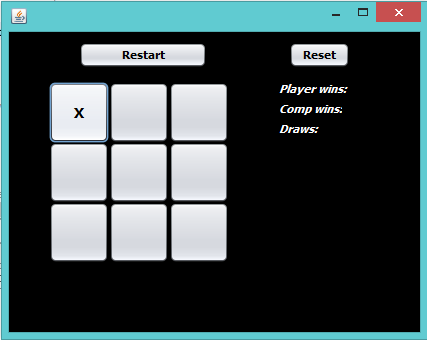
player to win (in which case the result is a draw).

When it is played on pen and paper, we think of various strategies based on current game state and decide our next move. The computer cannot intuitively discern the next move and hence must try various options before deciding on the best one.

Zobrist hashing helps us represent the game state in a compact manner so as to store each game state along with it's evaluation in a Transposition table which prevents the need for recomputing the best move to make for a given game state. We used NegaMax algorithm to decide the next move based on current game state.

Snapshot of the UI:





**ANALYSIS AND DESIGN OF UI:**

1) Play button, created using jButton, is used to open another frame which contains Tic Tac Toe board for the player to play.

2) Instructions button, also created using jButton, is used to open another frame which tells rules for te game.

3) We designed the Tic Tac Toe board using nine jButtons from jFrame class.

4) Restart button, on mouse click, opens the same jFrame with no X and O's. This is done by setText("X/O" // depending on the player) in the method jButtonXXMouseClicked(java.awt.event.MouseEvent evt) in Netbeans.

5) Reset button, on mouse click, sets all the scores back to 0. Done by setText method again.

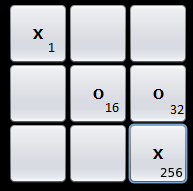
**IMPLEMENTATION:**

**1) Designing the board:** We took a 2X9 array, and initialized it.

**2)** Once, the game is started and it is in some random state, each of such game states are stored in as **transpositions** using **key** produced by **Zobrist Hashing** for that game state.

We used Zobrist hashing in the following way:

Say, current game state is like, below, and numbers represent values,



Part of the key holding X's state = 1^256

Part of the key holding X's state = 16^32

Our final key was a concatenation of these along with the current player whose turn it is.

Using this as the key, the evaluation of the next move to be made (obtained using Negamax algorithm) is stored in a HashMap to prevent reevaluation of transpositions.

3) **Designing board using UI:** jFrame class of java, gives many options. Board was designed using jButtons, jTextArea,jLabel,etc.

If Player 1 wins, his/her score is incremented by 1, else if computer wins, the Computer's score is incremented. If it is a draw, the draw count is incremented.

**CONCLUSION:**

Zobrist hashing is a very useful technique for implementing board games. We designed our project, assuming that the user plays optimally, and the computer always makes the optimal move, and there is NOT a single chance for a user to win, he can only make it a draw by the very nature of a game of tic tac toe played by two optimal players.

This implementation, compared to other board games is quite simple as the computer can look ahead to the full depth. What about chess? The computer can only look ahead to a certain depth, with 64 board positions and 12 types of pawns, and also assuming that the player is playing optimally. Zobrist hashing is very powerful tool in that situation.

This project could be taken further, by implementing the same Tic Tac Toe for bigger board configuration, say, 8X8 configuration or even 3-D 3X3X3 configuration.