Tic tac toe game computer science

Most of the research nowadays is focused towards problems that deal with complexity or are influenced by some kind of random events. Interesting about these problems is that if they are deterministic, then a solution is expected to exist, at least a theoretical one. These problems are often inspired by games, such as mathematical games (ex. Tic-Tac-Toe, Chess). On the other hand the point of randomness involved in these problems increases the difficulty of prediction on the possible solution, or in some situations outcome. This is thy, there are certain methods of operations devised, that in turn give some supplementary information to a decision maker. In most of the cases, the probability distribution of an even which took place randomly, it is possible to be affected by prior events.

These games are often played by at least 2 players (or many), out of which the one is called an opponent. The decisions at each step are made by the last move of the opponent. The operations research in these games is called game theory.

The vital Tic-Tac-Toe game consists of two players, X and O, who take turns marking the spaces in a 3A-3 grid (Crowley, 1993; Gardner, 1998).

The game usually begins with the X player, and the player who will manage to place three respective marks (in any direction, i. e. in a horizontal, vertical, or diagonal row) wins the game. This basic version of the game is rather simple, what allows the game to be used as a useful tool in combinatorial game theory, as well as a branch of artificial intelligence that deals with the searching of game trees (Beck, 2008). Using game theory there are few approaches that can be undertaken:

The game’s solution is resulted by dominance – when the game has only 1 rational strategy for each player

Minimax strategies decide a stable solution – useful if the opponent makes the wrong play

Minimax strategies do not decide a stable solution – using a probability distribution

Even though, game theory researches are made on the possible playing strategies, they might not be employed in real life when playing a game, because:

There might be too many strategies to enumerate (this number is simply too large to be estimated).

Players are not always rational.

There might be more than two players.

Real-life games are not zero-sum games.

This project deals with developing a Tic-Tac-Toe to be used on a mobile device. The following chapter discusses the Aims and Objectives of the game. Chapter 3 talks about a background research on this game, starting with a review on existing Tic-Tac-Toe games, which in turn leads to discussion about the existing models of this game and the proposed model of this work.

Finally, Chapter 3 concludes with a technology research concentrated towards Java 2 Platform, Micro Edition (J2ME). Chapters 4 and 5 describe the system requirement analysis and design on this work, and chapters 6, 7, and 8 include explanation on the implementation, testing and evaluation. And finally, chapter 9 concludes this work.

2. Aim and Objectives

The aim of this project is to develop a Tic-Tac-Toe game for mobile device. The game is supposed to consist of two parts, one a single player game (a player against a system), and the other a multi-player game (two players on their mobile devices, playing against each other). In order to accomplish these, the following objectives were defined.

Single player game

The player should play Tic-Tac-Toe game on his mobile device.

The player should have option to edit his name.

The player will start the game of choosing his symbol as X or O.

If player 1 selected X then O has to be automatically allotted to the mobile device as a second player, and vice versa.

The player has an option to choose the small game grid out of 4 small tic-tac-toe games.

If player X marked horizontally or vertically or diagonally of his symbol " X " in a row, then player X won that small match.

Finally, which player won the maximum small games will be declared as winner of the tic-tac-toe game.

Multi-player game

Using Bluetooth as communication channel the two players should play Tic-Tac-Toe game from different mobiles.

Players should have options to edit his name.

Once both players connected together, then first player will start the game of choosing his symbol as X or O.

If player 1 selected X then O has to be automatically allotted to player 2.

Then main game grid has to display in both mobiles.

Player2 have option to choose the small game grid out of 4 small tic-tac-toe games.

After grid selection both players will play tic-tac-toe game in that small grid.

If player X marked horizontally or vertically or diagonally of his symbol “ X ” in a row, then player X won that small match.

That small grid is marked with X and Player1 awarded 1 point, screen should zoom out and have to display whole main game grid and now player who won the previous game will have the choice to choose on which grid have to be select to play remaining game.

This process will be repeated until the whole Four (4) small games grids marked with X, or O, or T.

Finally which player won the maximum small games will be declared as winner of the tic-tac-toe game. then game ends.

3. Background Research

In this section the Tic-Tac-Toe game will be discussed in details. At the outset, the basic rules of the game are going to be covered. Then, there will be a review on existing Tic-Tac-Toe games, which in turn will lead to discussion about the existing models of this game and the proposed model of this work. Finally, this section is going to be concluded with a technology research concentrated towards Java 2 Platform, Micro Edition (J2ME).

3. 1 Basic Rules of Tic-Tac-Toe game

The basic Tic-Tac-Toe game consists of two players, X and O, who take turns marking the spaces in a 3A-3 grid (Crowley, 1993; Gardner, 1998). The game usually starts with the X player, and the player who will manage to place three respective marks wins the game. The marks can be placed in any direction, i. e. in a horizontal, vertical, or diagonal row. This basic version of the game is rather simple and very often leads to draw. This simplicity allows the game to be used as a useful tool in combinatorial game theory, as well as a branch of artificial intelligence that deals with the searching of game trees (Beck, 2008).

The Roman Empire is known to have established the beginnings of the earliest known variant of tic-tac-toe. It originated around the first century BC

(Crowley, 1993). At that time, the game was called Terni Lapilli. Instead of having any number of pieces, each player only had three. The game was played by moving them around to empty spaces to keep playing. However, according to Claudia Zaslavsky’s book, the game Tic Tac Toe is originating from ancient Egypt (Zaslavsky, 1982).

Chess and Tic-Tac-Toe are one of the most famous games to which the moves are not left to chances, rather than pure mathematics and logical reasoning. In these games, a player wins by achieving a winning configuration first, like for instance: checkmate in chess, and 3-in-a-row in a basic Tic-Tac-Toe game in 3? 3 board (Gardner, 1998). Thus, the question which can be posed at this point is: How a player can achieve a winning configuration first? Even though there isn’t a general theorem to answer this question, there might be a well-known strategy stealing argument that can give a partial answer about when a player can achieve a winning configuration first (Beck, 2008).

In order to find a winning strategy, in theory all the paths could be explored. However, in practice this is not easy because the total number of strategies can be calculated a double exponential function of the size of the board. For example, a 3-dimensional 5A-5A-5 version of Tic-Tac-Toe, has about 3125 positions. This is because each one of the 53 cells has 3 options:

Marked by the first player,

Marked by the second player, or

Unmarked.

Thus the backtracking on a graph of 3125 vertices takes at least 3125 steps. This is the main reason that this 3-dimensional 5A-5A-5 version of Tic-Tac-Toe remains unsolved up to date. Moreover, only two explicit winning strategies are known from in the whole class of

nA-nA-· · ·A-n = nd Tic-Tac-Toe games. This is the 33 version and it is characterized with an easy winning strategy, and the 43 version that in turn has an extremely complicated winning strategy.

In order to play a perfect tic-tac-toe game, i. e. a win or a draw, the player can play given they move consistent with the uppermost possible moves. This is presented in the following table (Crowley, 1993):

Win

If the player has two in a row, play the third to get three in a row.

Block

If the opponent has two in a row, play the third to block them.

Fork

Create an opportunity where you can win in two ways.

Block opponent’s fork

Option 1: Create two in a row to force the opponent into defending, as long as it doesn’t result in them creating a fork or winning. For example, if “ X” has a corner, “ O” has the center, and “ X” has the opposite corner as well, “

O” must not play a corner in order to win. (Playing a corner in this scenario creates a fork for “ X” to win.)

Option 2: If there is a configuration where the opponent can fork, block that fork.

Center

Play the center.

Opposite corner

If the opponent is in the corner, play the opposite corner.

Empty corner

Play in a corner square.

Empty side

Play in a middle square on any of the 4 sides.

Initially, the player that starts first gets the “ X” and has 3 probable positions to mark in his turn. Even though it seems that there are 9 possible positions, as there are 9 squares in the grid, by rotating the board, this is not the case. It can be observed that:

Every corner mark is tactically equal to every other corner mark, and

Every edge mark is tactically equal to every other edge mark.

There are therefore only three possible first marks: corner, edge, or center. The first player could win (or make a draw) from any of these starting marks. It can be also observed that playing a corner would give the opponent the smallest choice of squares. This is a nice strategy as could be played to avoid losing (Zaslavsky, 1982) .

The second player can be identified as “ O” and this player must respond to X’s opening mark. However, this should be done in such a way as to avoid Player X to win. It can be stated that Player O must always respond with (Zaslavsky, 1982):

To a corner opening with a center mark,

To a center opening with a corner mark and

To an edge opening either with a center mark, a corner mark next to the X, or an edge mark opposite the X.

Any different play would allow X to compel a win. After every next turn of player X, the player O should follow the above list. This way the player O can achieve a draw (or a win if the player X makes a weak play).

3. 2 Existing Tic-Tac-Toe games

As many other games like: three men’s morris, nine men’s morris, pente, gomoku, Qubic, Connect Four, Quarto and Gobblet, Tic-Tac-Toe also has the same goal, i. e. a player wins if he is the first one to get n-in-a-row. Basically, if a generalization is to be provided, it can be concluded that all the different formations of Tic-Tac-Toe can be represented as nd-games, which are accordingly played on a d-dimensional boards with edge n (Zaslavsky, 1982). As it was discussed in the previous section as well, the original Tic-Tac-Toe game is actually a 32-game.

There are many variations, discussed as follows (Patashnik, 1980; Gardner, 1998; Beck, 2008).

A slightly different version of a Tic-Tac-Toe game is the 33-game, played on a 3x3x3 board (Patashnik, 1980).

It can be noted that this game gives good opportunities to the player that plays first, so he could achieve an easy win by playing at the center with his first move. Similarly, playing on a 4x4x4 board also gives the first payer better chances for wining.

More complex version of a Tic-Tac-Toe game is playing it on a board with higher dimensional space. 4 dimensional, i. e. 3A-3A-3A-3 board is one of the most commonly played Tic-Tac-Toe (Patashnik, 1980).

In this version there are 2 possible aims. One of them is to position elements through all of the board, thus the player that has more rows of 3 totally than the other one is the winner of the game. And the other strategy is to include 4 players, in which case the winner is the payer that will get a row of 3 first.

Another version is the misere tic-tac-toe game. It is played according to its conventional rules, such as in this variation 3? 3 game would be a draw, whereas the winner is the player that will get n in a row (Berlekamp, 1982).

Quite a new game is the Tic Tac Tactic variation of tic-tac-toe (Berlekamp, 1982). This game is played on a 3 dimensional curved board, and the here each player tries to roll a ball at least half the way, as it would then drop on a grid that has 9 positions (3? 3 grid). This way the players should make a row of 3 in order to gain a ball. The winner is the player that will have won the first 5 balls. In order to roll their balls precisely, they could use a device that helps into changing a ball's trajectory.

Yet another version is the nine board tic-tac-toe. In this game, there are in essence 9 boards, arranged as 3? 3 grids, and the first payer can start on any of them by his choice (Gardner, 1998). The following moves are supposed to be places on the board chosen by the first player. Once this board gets full and there is no more space left, the next move can be again on any of the boards left, by the choice of the player. The winner is the one that will achieve 3 in a row. However, having 9 boards gives the game yet another spirit than the usual tic-tac-toe game, as the players can have an opening, middle and end of their game.

Similar to the nine board tic-tac-toe game is the super tic-tac-toe game (Beck, 2008).

The difference in this variation is that this game does not end once a player makes 3 in a row in one of the 9 boards. As an alternative, the position of that board is marked on a new 3? 3 grid, and the winner is the one that will make 3 in row there.

Tic-Tac-Chess is an interesting combination of games, as it involved playing a chess game, as well as a tic-tac-toe game at the same time (Beck, 2008).

In this variation, once a player captures a piece from the challenger on the chess game, makes a move on the tic-tac-toe game (even if the challenger has not placed anything on the tic-tac-toe game yet). And of course, the winner is the player that will make 3 in a row on the tic-tac-toe game first.

A game that in essence is an isomorphic to a tic-tac-toe game, even though it seems as a completely different game, is described as follows (Beck, 2008). Basically, there are 2 players that should say a number between 1 and 9, without repeating the previously said numbers. The winner is the player that will first make a sum of 15. This game is isomorphic to a tic-tac-toe, because if those numbers are to be placed on a 3? 3 magic grid, then it will be exactly as playing a tic-tac-toe game, because a straight line is formed only if the sum of the numbers is 15. This information is mostly useful in programming variations of a tic-tac-toe game.

Another different variation again employs numbers from 1 to 9 (Gardner, 1998). These are to be placed on a 3? 3 grid, but must be held with an order of precedence defined by the players. Then the players play a tic-tac-toe game, filling the grid by the precedence defined beforehand.

Check Lines is a very old variation of tic-tac-toe game, invented in the 1970s by Tri-ang Toys & Games. In this game the board is actually any geometrical pattern that consists of 12 lines.

There are 11 holes in total, distributed in a way that each line has 3 holes. At this point, each player is given 5 coins, and each player on their turn should place a coin on the board. The winner is the one that will have first completed 2 lines. Because the players have only 5 coins, this means that they have to complete intersecting lines. If none of the players have won after placing their 5 coins, then they will continue playing by replacing the position of the coins, on the remaining spaces, with the rule that it must be done only on an adjacent hole.

Very similar game to the tic-tac-toe game is the Toss Across game. Here, the players are given bags with beans and they are throwing them on a big board for marking the squares.

Star Tic Tac Toe is another popular variation of tic-tac-toe. This game is played with checkers like movable pieces. It has a 3? 3 board, thus a player has 3 pieces accordingly.

The participants keep on replacing pieces into the spaces which are left empty in the board, until one the players wins; this actually adds some more dynamism in the game. Moreover, the players have supplementary star shaped pieces, which can be swapped.

Similar category of games as the previous bullet, are the: Mojo, Mojo Too and Mojo tic-tac-toe games. In these variation the payers also pieces and pawn(s) onto empty positions until there is a winner.

Moreover, there are many shows based on the tic-tac-toe game, as well:

Hollywood Squares is a show with 9 celebrities, which fill the cells of the tic-tac-toe grid.

Tic-Tac-Dough is a show on which the players put symbols up on the board. This is achieved by answering queries in a variety of categories.

In Beat the Teacher competitors respond to questions to win a turn, again on a tic-tac-toe grid.

On The Price Is Right, there is a pricing game called “ Secret X”, in which players must estimate prices to win X’es, in order to place them on a blank board. They must position the X’es as to provide speculation of the location on the “ secret X”. This is in turn hidden in the middle line of the board, forming a tic-tac-toe line across.

The fictional game D’ni game of Gemedet, has an aim to place 6 balls in a row to a 9x9x9 grid (Gardner, 1998).

The fictional game Squid-Tac-Toad, has an aim to place 4 or 5 balls in a row to a 4? 4 or 5? 5 grid, accordingly (Gardner, 1998).

A more simplistic variation of this game is having the rules as of the “ Y” formations to count as a win. This is rather simple, because all the scenarios basically forming some kind of a “ Y” configuration.

Quantum tic tac toe is yet another variation in which the participants are positioning a quantum superposition of numbers on a tic tac toe board (Gardner, 1998).

A larger grid (for example 10? 10) tic-tac-toe games also exist. In a 10? 10 grid the winner should place 5 in a row. The more the grids there are on a board, the larger complexity of the game is.

Another similar game named Go-moku, originating from Vietnam, also has the strategy for a player to get 5 in a row in order to win the game (Gardner, 1998). The players put X’s and O’s, but in order to try blocking each other, in this variation they should also try to create changes for wining. Another difference is that the board has no limit, thus the game is played until there is a winner.

Three Men’s Morris and Nine Men’s Morris are also variations, in which there is a limiting on the number of pieces in order for a move to be allowed (Gardner, 1998).

Finally, the last variation of the tic-tac-toe game, employs the words: “ eat, an, laf, it, line, if, lot, on and foe”. In this game, the winner is the one that will select 3 words that start with the same letter. If the game was places on a tic-tac-toe grid, it would mean 3 words in order to form a line (three in a row line).

3. 3 Proposed model

There are quite a few algorithms hat can be used for creating the Tic-Tac-Toe’s game strategy. The most popular ones are the semantic algorithms and the lexical algorithms. For this project, a lexical algorithm was utilized. The model of the tic-tac-toe game described in this work contains 2 different game strategies. Basically, the one strategy is the Single Player game where a player plays against a system. The other strategy involves Multiple Player environment, and it is being played by a player versus another player.

In order to analyze this game, a decision tree might be used. Moreover, for the analyzing part it should be assumed that both the players in the Multiple Player environment, and the single player in the Single Player game, are in essence experienced. This means that the result of a game can be foreseen after the first move from each participant (again assuming that there are no mistakes). Let us represent with 1 if the player that has the X wins and with -1 if the player that has the O wins. The following figure represents the decision tree after the first move from each participant. As it was already discussed in section 3. 1 Basic Rules of Tic-Tac-Toe game, the tic-tac-toe game is symmetric and therefore it is sufficient to consider only the squares 1, 2 and 3 for the first player (see the figure below). The rest of the moves are symmetric and will be presented. So, following this reasoning, the first player has the positions 1, 2 and 3 available, and the second player has the remaining two positions.

The figure above presents an expansion, so called an extensive form. It demonstrates that even in the simplest scenario the decision tree can be quite large. For example, if the first two moves were to be presented, this would be impossible to be demonstrated on a single page.

Similarly to this discussion, the strategic form of the game can be presented by a different model, i. e. as a matrix. In order to demonstrate this approach, it should be assumed that the players choose one strategy and they strictly follow it when their turn comes. Of course, each strategy should represent all the paths of action and in every possible situation.

At the beginning, let us assume that there is a strategy that the first player uses for their first move, and another strategy for the first move of the second player. This logic would create some rules like the following (Zaslavsky, 1982):

For the first player: select one of the nine squares on the game board.

For the second player: Select one of the nine squares on the game board. If the first player already uses the selected square, then

aˆ? put an O in square 3, 5, 7, or 9 if an X is in square 1 (center)

aˆ? put an O in cell 1 if an X is in cell j.

These rules are examples of complete strategies, and these can be selected by the payers before the beginning of the game, and thus followed with their first moves. The strategic form of a tic-tac-toe game is presented on the figure below. It should be noted that the entries in the table below are in essence the values of the game. They hold values for every possible selection of strategies.

Each tic-tac-toe game that can be actually presented in an extensive form would have an equivalent strategic form similar to the one shown in the table presented above. Moreover, this table is also equivalent to the matrix established previously. The payoff matrix in cooperation with the descriptions of the strategies comprises the model for the two-person tic-tac-toe game.

3. 4 Comparison of Proposed model with Existing Models

The semantic algorithm is yet another approach towards the tic-tac-toe game. The semantic algorithm is in essence a learning algorithm, and it might be structured in the following way. It might have as initial information the ability to recognizing the 3 states of a game: lost, won or a draw. The algorithm in this case would play the X, and it will play against another algorithm, i. e. the O. As soon as a game is finishes, the information if the game was won or lost is stored. Moreover, the moves are presented with the smaller letters “ x” and “ o” accordingly. A possible structure of stored information could be the following line: x5 o3 x9 o4 x1 won. The first move is always randomly selected. So, given that the algorithm played 7 (x7), and the opponent played 6 (o6), the algorithm will search for previous games that are most similar to x7 o6. If such a case is found, then the following rules apply:

If the game found was a win, than the algorithm will try to reproduce the move. If the position is not available, it will play randomly.

If the game found was a loss, the algorithm will try to correct the move, by not placing an element in the same position as in the lost game.

This is repeated until there is a winner. Moreover, if a game end with a draw, it is not saved in the database.

Comparing this algorithm with a lexical algorithm such as our proposed model, it might be noted that the semantic algorithm usually plays very badly at the begging. But, after a certain number of games, the learning curve of the algorithm becomes better. On the other hand, our proposed model behaves well during all the stages of the game.

3. 5 Technology Research (j2me)

Being quite different from other programming languages, Java does both compiling and interpreting when it comes to process code.

As it can be seen from the photo above, the source code (i. e. the . java files) is initially translated by the compiler. This gives an output of an intermediate language, called Java bytecode (i. e. the . class files). The bytecode is then ready to be executed (or in other words, interpreted) within a particular virtual processor, known as the JVM (Java Virtual Machine) (Hayun, 2009; Knudsen, 2008). This is in essence a simulated processor that executes all the bytecode commands. The Java Virtual Machine is the basic components that give to Java the feature to compatibility. This is simply because it represents a reliable layer between bytecode and the concrete machine instructions, translated at runtime.

Over the years, the Java language has undergone many changes and development. J2SE (Java 2 Standard Edition) had its first edition targeting GUIs, applets, and other basic and rather simple applications. Recently, the language was extended with the Java suite known as J2EE (Java 2 Enterprise Edition). This edition is based for server side development, and includes tools for: database access, messaging, content rendering, inter-process communications, and transaction control (Hayun, 2009; Li, 2005). J2ME (Java 2 Micro Edition) came into existence as to cover the needs for applications targeting mobile devices. As it can be seen from this short overview, there are versions of Java to suit different environments: from the enterprise development tools intended for use in servers, to the micro systems. An important thing to note at this point is that the separation between platforms is not just unconditional (Knudsen, 2008). Many times these are not a simple line than can be drawn. In order to demonstrate this, it might be explained that Java 2 Micro Edition development sometimes requires the use of Java 2

Enterprise Edition and Java 2 Micro Edition. This is the case with multiplayer games for instance, so and Java 2 Micro Edition is used for the client side, but Java 2 Enterprise Edition is used for the server side of the application/game. Moreover, different Java editions target different hardware configurations. Similarly, there are 3 virtual machines to be used for the different environments (Li, 2005). For example, Hotspot VM is a default virtual machine suitable for a executing the full-scale edition of JavaHotspot. JavaHotspot is a newer type of virtual machine competent of vigorously optimizing a great deal of executed code (called as hotspots) during the runtime (Li, 2005). Other versions of virtual machine are the Compact Virtual Machine (CVM) and Kilobyte Virtual Machine (KVM). These are in essence smaller virtual machine implementations. They are targeted to run within the restrictions of the limited resources found on the micro devices (these will be discussed later in this section, as well).

The requirement of having another version (like the Java 2 Micro Edition) for the mobile devices came because these devices do not have sufficient recourses to run Java 2 Standard Edition, since J2SE was clearly way excessively large to fit on even the bigger micro devices. However, the question was imposed initially was which features should be left out from the J2SE, so to be minimized in a smaller edition. Also, having great diversity of different devices, it would not have been a nice decision to restrict all the J2ME applications to the lowest compatible hardware configuration (Li, 2005; Kochnev, 2003). Moreover, this solution would not have been practical as well, because it would incorrectly neglect the capabilities of the higher end devices. The final solution is comprehended through a mixture of J2ME configurations and profiles (Krikke, 2005). It represented a revised Java architecture, which actually offers for the leaving out of parts of the platform, at the same time as addition to device and category precise components. Along these lines, the configuration would identify the abilities of a Java platform intended for use on a sequence of analogous hardware. Possible components that can be removed are the following (Kochnev, 2003; Lefevre, 2005):

Java language mechanism

smallest amount hardware necessities, such as the memory, screen size, and processor power for the family of devices

integrated Java libraries

By utilizing this approach, there are actually 2 preset configurations for mobile devices: one for somewhat restricted devices such as PDAs and Set-Top-Boxes (for instance the digital TV receivers), and another one for devices such as pagers and mobile phones.

These two configurations are (Kochnev, 2003; Krikke, 2005; Lefevre, 2005):

CDC (Connected Device Configuration)

CLDC (Connected, Limited Device Configuration)

All of these configurations are to be reviewed as follows. On the other hand, a good example of java profiles is the UI (User Interface) for mobile phones. For example, the J2ME configuration CLDC that wraps this type of device, keeps out the typical Java UI libraries (AWT and Swing). The devices do not have the ability of presenting anything derived from these libraries in any case. This is due to the fact that their screens are just too small. Thus, there is no point to slaughtering valued space on them. The solution was to generate an innovative User Interface, fitting to the exact necessities of the poor mobile's LCD display. The consequential LCD UI is built-in in the CLDC profile. This targets MIDs (Mobile Information Devices), for this reason the name is MIDP.

The CDC is built for bigger devices such as digital TV set-top-boxes and PDAs. These are devices characteristically with numerous space of memory. The CDC is the bigger brother of the J2ME configurations. It encloses a single profile (the Foundation profile) as well as a high performance virtual machine (known as the Compact Virtual Machine – CVM). This Java language implementation, as well as the API, practically has all the influence of J2SE.

Unluckily, the CDC is not accessible on the platform for the most micro-game players (the mobile phones).

The CLDC is especially targeted to micro devices, like mobile phones. It fundamentally defines a standard, which in turn is used by all the device manufacturers that put into service a Java run-time environment. This enhances third-party development, because this same standard is being utilized. The CLDC configuration was developed as part of the Java Community Process (JSR-30) (Krikke, 2005; Lefevre, 2005). It affects many aspects of Java development and delivery, which include (Knudsen, 2008; Li, 2005):

Target device distinctiveness

The security form

Application administration

Language dissimilarity

JVM variations

Built-in class libraries

Talking about the security model, the J2SE’s offered protection system was too big to fit within the restrictions of the CLDC target platform. It might be stated that by itself it would probably have gone beyond all accessible memory (Knudsen, 2008). Therefore, the modified model removed many of the functionalities. This in turn requires far less resources. But nevertheless, this overview made it greatly simpler for wrapping all the fine points. Basically, there are two major parts to the CLDC security model (Li, 2005; Kochnev, 2003).

Virtual machine security

Application security

The modified security model for the CLDC, also provides some significant foundation for the application execution models. The objective of the virtual machine protection layer is to defend the underlying device from any harm that an executable code could probably do. With the usual conditions, a bytecode verification process is performed. This should be done preceding any code execution. This verification procedure basically authenticates a class-file bytecode. Thus in turn ensures that the code is safe for execution.

Very important here is that incorrect instructions or corrupted memory outside the Java environment, will be eliminated and protected, as well. This process is very light as well. It only consumes 50 KB of code space additionally to 100 KB of heap. However, one might argue that even though light and insignificant on larger systems, it is still a good space when mobile devices are taken into consideration. Taking into consideration the notion of Application Security, the class-loader verification procedure (mentioned in this section) is very much limited as well.

4. System Requirement Analysis

At the beginning of the game, the user should have the ability to select between a single player or a multiple player mode.

Choosing a single player mode, gives the user the ability to play first and to select where he wants to play his move. As soon as the user marks the box where he would like to have positioned his move and presses Ok, there should be an ‘ X’ placed on that position for him. After that, the system should play an ‘ O’ and then give the turn back to the user again.

After a win is achieved, the system should show where the win took place, and should display statistical results (score board) about the played games so far and their results.

After a player wins 3 games, the game stops and the following screen should be displayed:

This screen should show whether the user or the software won the game. Important to mention is that the 1st game is assigned to the player to play first, the 2nd game to the software to play first, and so on.

The multi-player game has the following requirements. Initially a player should have the opportunity to assign his name:

Then, he should choose whether he is hosting a game, or joining an existing game.

If the player is hosting a game, this will start the server and the system will now wait for an incoming connection. On the other hand, if the player is joining a game, this will start the application which will be searching for an existing Tic Tac Toe server in range.

Once the 2 players are connected, the rest of the game will be like in a single player game, just that now instead of the application as a second player will play a person.

5. Design

5. 1 Proposed algorithm

instead of the

5. 2 Information how we design this model

6. Implementation

This project consists of 3 packages: game, logic and screens. This java documentation (Javadoc) was created as well.

The game call consists of: BluetoothManager, BluetoothManagerServer, DrawAnimation, MultiplayerWinningAnimation, TicTacToeGameEngine, TicTacToeMIDlet and WinningAnimation. The relationship between these classes is presented below:

7. Testing8. Evaluation

Having presented the system requirements analysis, implementation and testing, one can now conclude that this application of a Tic-Tac-Toe game behaves according the initial requirements. Moreover, after having presented this software, there might be few answers left to analyze. Basically, by running the tests one can now answer:

How complicated is the game?

How much benefit has the player that plays first?

How much luck is needed in the game, or in other words, can this game be played by chances?

In order to examine how complicated this game is, one should take into consideration the number of possible games. Given that the player that plays first can make any of 9 moves, the second any of 8 remaining, and so on, there are total 9! = 362880 possible games. On the other hand, given that a game can end before the ninth move or some games are symmetrical, could lead to an approximation of 255168 possible games. Comparing this number with other games, one can now conclude that 3? 3 Tic-Tac-Toe game has medium complexity.

Talking about how much benefit has the player that plays first, one might answer this question with the sub-question about how frequent the first player wins, and given that both the players are interchanging who plays first evenly. Testing this question with the software, one might conclude that there are 2 possible answers to this question:

If the two players play completely random game

If the two players play a perfect game

Once the two players who play completely random game, the first player wins 58. 49% of the time, and the second player wins 28. 81% of the time. This leaves the game as a draw 12. 70% of the time. This is can be translated to a 64. 84% – 35. 16% advantage. I calculated these results based on this software, adding a code that enumerates all the possibilities.

The second choice, when two players play a perfect game, the game is always a draw. Basically, there are always 3 answers (either the 1st player always wins, the 2nd player always wins, or the game is always a draw) in a deterministic game. On the other hand, games like backgammon are non-deterministic games, and there one can have different answers as well.

And the last question that can be observed by this software is whether this game can be played by chances and how much are these changes. In order to be able to answer this question, one can actually formulate a sub question: How often a perfect player will win when playing against a random player? But what is a perfect player? It is assumed that a perfect player always assumes that his opponent is also perfect – in other words, when 2 moves both lead to an identical result with perfect play, the software cannot differentiate between them. A perfect move chooses the move which offers the most chances for the opponent to make an error (for example playing against the system). The results are presented as follows:

When the random participant moves first, the random player wins 0. 00% of the time. On the other hand, the perfect player wins 80. 64% of the time. The game is drawn 19. 36% of the time. Interesting here is that this is corresponding to a 90. 32% – 9. 68% benefit.

And the other case is when the perfect player moves first. Here again the random player wins 0. 0% of the time. However the perfect player wins 99. 47% of the time. And finally, the game is drawn 0. 53% of the time. This is corresponding to a 99. 74% – 0. 26% benefit.

Uniting these two results, it can be calculated that a perfect player enjoys a 95. 03% – 4. 97% advantage.

9. Conclusion

This project discussed developing a Tic-Tac-Toe game for mobile device. The game consisted of two parts, one a single player game (a player against a system), and the other a multi-player game (two players on their mobile devices, playing against each other). In order to develop the Tic-Tac-Toe game correctly, this work followed an object oriented approach (the waterfall model). The system requirements, analysis, design, implementation and testing were discussed in details. Moreover, this work consists of a background research, as well. In this section the logic behind the Tic-Tac-Toe game was discussed in details. At the outset, the basic rules of the game were presented. Then, there was a review on existing Tic-Tac-Toe games, which in turn lead to discussion about the existing models of this game and the proposed model of this work. Finally, the background research was concluded with a technology research, concentrated towards Java 2 Platform, Micro Edition (J2ME). All this work resulted in a nice and user friendly Tic-Tac-Toe game for mobile devices.