# Tigers and Goats
# A Board Game for Android

P. Shouthiri*

*Department of Mathematics, Eastern University, Sri Lanka.
shouthiris@gmail.com

***Abstract***- Mobile application development is one of the recent trends in computing Industry. Android is one of the largest platforms around the world that run in several smart phones and tablets from various manufacturers like Google, Motorola, Samsung, HTC etc. This paper aims to develop a board game for android platform, exploring a new dimension in the traditional board game to make it more interesting and challenging. It brings fun and simplicity of 'Tigers and Goats' game with new features. It includes computer controlled intelligent opponents whose aim is to challenge the human player. 'Tigers and Goats', is a simple game application in android targeting people to teach how to make correct and usable models in a relatively short amount of time. The application presents a graphical user interface with triangle shaped game board and invisible buttons on the game positions. The application allows the user to place pieces on game positions by touching on it. The user's goal is to take out ten goats or tied down all tigers. The application contains two-player mode and single-player mode. The game itself is core strategy. It requires a fair understanding of the principles of attack, defense, and teamwork, and is suitable for players of the ages six and up. While it takes just a few minutes to learn, it can take ages to master. This will be a challenging game, and will test your strategy skills.

***Index Terms*** -Android, Board Game, Game Application, Graphical User Interface.

## I. INTRODUCTION

One of the most widely used mobile OS these days is 'Android'. Android was founded in Palo Alto of California by Andy Rubin, Rich miner, Nick sears and Chris White in 2003. Later Android Inc. was acquired by Google in 2005. Over the years, there have been multiple versions released each with its own name. Since the initial release of the Android operating system, there have been more than 14 versions released up until the latest version, 7.0 Naugat. Each major version of Android has a dessert-based nickname, and they are all in alphabetical order.

Initially, the deployment target for Android was the mobile-phone arena, including smart phones and lower-cost flip-phone devices. However, Android's full range of computing services and rich functional support has the potential to extend beyond the mobile-phone market. Android can be useful for other platforms and applications. Android applications are written in java programming language, which is open source software and the Android SDK is available to developers free.

Nowadays people are interested in playing games on their mobile devices, especially when they travel a distance. One of the most interesting type of game is 'Tigers and Goats' is a board game for Android based smart phones, which gives enjoyment and helps increasing the player's strategy skills.

'Tigers and Goats' is a strategic, two-player board game that originates in South India named as Aduhuli. This game between two opponents, whom we call "Tiger" and "Goat", is asymmetric game in that one-player controls three tigers and the other player controls up to fifteen goats.

As said above, the proposed application is an Android strategy game targeting people who wish to learn to play board games on mobile devices and making correct moves to win. The intention is also to popularize this traditional game as an Android app, in order to recall our traditional game being played by younger generation for long time. Android is chosen because of its wider popularity among mobile-phone users.

## II. TIGERS AND GOATS GAME

Tigers and Goats (Aduhuli) is a strategic, two-player board game that originates in South India. The game is asymmetric in that one player controls three tigers and the other player controls up to fifteen goats. The tigers 'hunt' the goats while the goats attempt to block the tigers' movements. This game is also seen in Nepal with different board but the rules are same. The game is played on a five by five-point grid or triangle shaped board. Pieces are positioned at the intersection of the lines and not inside the areas delimited by them. Directions of valid movement between these points are connected by lines.

The game play takes place in two phases. In the first phase, the goats are placed on the board while the tigers are moved. In the second phase both the goats and the tigers are moved. For the tigers, the objective is to 'capture' ten goats to win. Capturing is performed by jumping over the goats, although capturing is not obligatory. The goats win by blocking all the tigers' legal moves. It has many similarities to the Nepal game Bagh Chal (goat-tiger game), though the board is different.

### *A. Rules of the Game*

At the start of the game all three tigers and all goats are start off the board. The pieces must be put at the intersections of the board lines and moves follow the lines. Tigers play first. In a placement phase, tigers place one on each move, on any empty spot. After each placement of a tiger, a goat will be placed. After the placement of three tigers, tiger-player moves one of his/her tigers according to either of the following two rules:

1) A tiger can slide from his current spot to any empty spot that is adjacent and connected by a line; or
2) A tiger may jump in a straight line over any single adjacent goat, thereby killing the goat (i.e., removing the goat from

the board), provided that the landing spot next to the goat is empty.

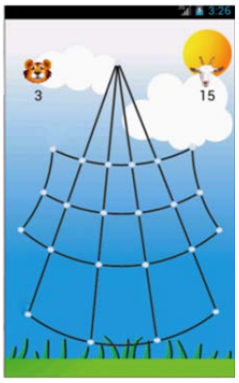

Figure 1. Initial game position of Tigers and Goats

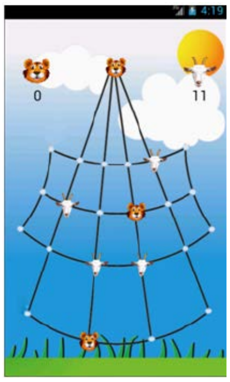

Figure 2. After the placement of three tigers

If tigers have no legal move, tiger-player loses the game (see Figure 3); if a certain number of goats have been killed (typically ten), goat-player loses. If neither of these conditions ever arises, the outcome is a draw by repetition.

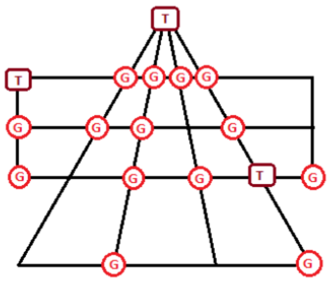

Figure 3. Tiger player loses the game

After the placement of all goats (and tigers have already been placed now), goat-player has to move on his/her turn to play, goat-player must slide any of surviving goats to an adjacent empty spot connected by a line. If there are 10 or fewer goats on the board, three tigers cannot have blocked by all goats and a move for tigers is always exists. In exceptional cases, which arise only if goats cooperate with tigers, the three tigers can surround twelve or more goats in a corner and prevent any goat moves. In such rare scenario where goats have no legal moves, goats lose the game.

## B. *Game features*

1) Play Tigers or Goats in the single player mode against the Android device (Human-Android or Android- Human).
2) Player can choose role to play (Tiger/Goat).
3) Two players can also play against each other on the same Android device (Human-Human).

## C. *GUI Design*

This application was implemented on eclipse using Java and XML, and also developed using Android 4.0 - API level 14(Ice Cream Sandwich), because of its support for Grid Layout and Drop Down actions. This application is compatible with devices running API level 14 and higher versions. Random moves and onClick event are used to find the location for the tiger's, goat's placement and moves.

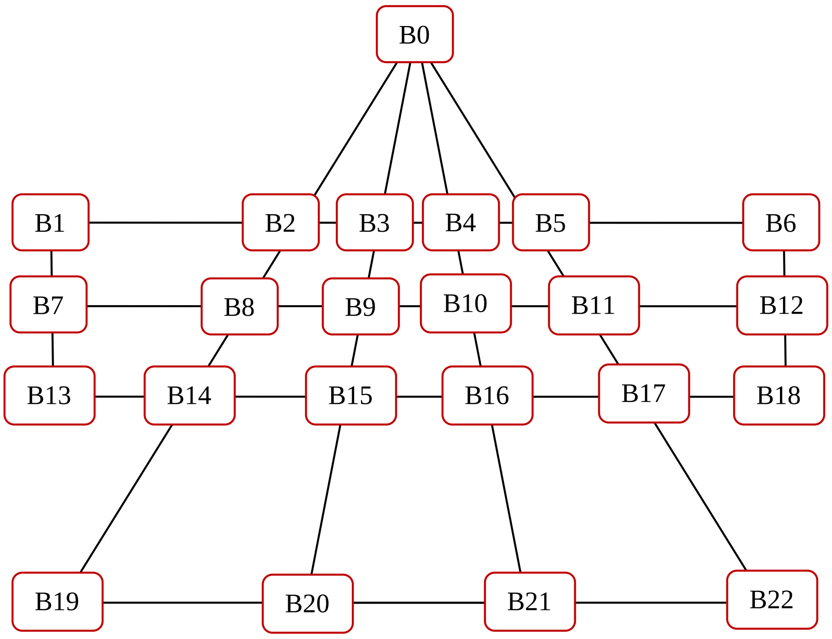

Figure 4. Name of the board positions to place the pieces

## III. ALGORITHM

The game board consist 23 spots named as B0 to B22. Human player can choose a role as tiger/goat. Assume that the human-player is playing goat as G and the Android is playing tiger as T.

### A. *Check for winning condition*

At the start of its turn, the Android first checks for tigers winning cases and goats winning cases. For the tiger's winning, the number of G in the board is less than or equal five. For the goat's winning, the moves for all three T are null.

Algorithm:

Let B[i] be the i$^{th}$ spot. So I lies between 0 and 22. Let numerically T be represented by 0 and G by 1 and empty spot by 2.

```
value=0
For (int i=0; i<22; i++) {
   If(i==0) {                    //Tiger blocked in spot B0
      If(B[i] == 0 && B[i+2] == 1 && B[i+3] == 1 &&
      B[i+4] == 1 && B[i+5] == 1 && B[i+8] == 1 &&
      B[i+9] == 1 && B[i+10] == 1 &&  B[i+11] == 1) {
         value=value+1
      }
   }
   If(i==1) {                    //Tiger blocked in spot B1
      If(B[i] == 0 && B[i+1] == 1 && B[i+2] == 1 &&
      B[i+6] == 1 && B[i+12] == 1) {
         value=value+1
      }
   }
   If(i==2) {                    //Tiger blocked in spot B2
      If(B[i] == 0 && B[i+1] == 1 && B[i+2] == 1 &&
      B[i-1] == 1 && B[i-2] == 1 && B[i+6] == 1 &&
      B[i+12] == 1) {
         value=value+1
      }
   }
   If (i==3 || i==4) {           //Tiger blocked in spot B3 or B4
      If(B[i] == 0 && B[i+1] == 1 && B[i-1] == 1 &&
      B[i+2] == 1 && B[i-2] == 1 && B[i+6] == 1 &&
      B[i+12] == 1 && B[i-i] == 1) {
         value=value+1
      }
   }
   If(i==5) {                    //Tiger blocked in spot B5
      If(B[i] == 0 && B[i+1] == 1 && B[i-1] == 1 && B[i-2]
      == 1 && B[i+6] == 1 && B[i+12] == 1) {
         value=value+1
      }
   }
   If(i==6) {                    //Tiger blocked in spot B6
      If(B[i] == 0 && B[i-1] == 1 && B[i-2] == 1 && B[i+6]
      == 1 && B[i+12] == 1) {
         value = value+1
      }
   }
   If(i==7) {                    //Tiger blocked in spot B7
      If(B[i] == 0 && B[i-6] == 1 && B[i+6] == 1 &&
      B[i+1] == 1 && B[i+2] == 1) {
         value =value+1
      }
   }
   If(i==8) {                    //Tiger blocked in spot B8
      If(B[i] == 0 && B[i-1] == 1 && B[i+1] == 1 &&
      B[i+2] == 1 && B[i-6] == 1 && B[i-i] == 1 && B[i+6]
      == 1 && B[i+11] == 1) {
         value= value+1
      }
   }
   If (i==9 || i==10) {   //Tiger blocked in spot B9 or B10
      If(B[i] == 0 && B[i-1] == 1 && B[i-2] == 1 && B[i+1]
      == 1 && B[i+2] == 1 && B[i-6] == 1 && B[i-i] == 1
      && B[i+6] == 1 && B[i+11] == 1) {
         value=value+1
      }
   }
   If(i==11) {                   //Tiger blocked in spot B11
      If(B[i] == 0 && B[i-1] == 1 && B[i+1] == 1 && B[i-2]
      == 1 && B[i-6] == 1 && B[i-i] == 1 && B[i+6] == 1
      && B[i+11] == 1) {
         value=value+1
      }
   }
   If (i==12) {                  //Tiger blocked in spot B12
      If(B[i] == 0 && B[i-6] == 1 && B[i+6] == 1 && B[i-1]
      == 1 && B[i-2] == 1) {
         value=value+1
      }
   }
   If (i==13) {                  //Tiger blocked in spot B13
      If(B[i] == 0 && B[i+1] == 1 && B[i+2] == 1 &&
      B[i-6] == 1 && B[i-12] == 1) {
         value=value+1
      }
   }
   If (i==14) {                  //Tiger blocked in spot B14
      If(B[i] == 0 && B[i-1] == 1 && B[i+1] == 1 &&
      B[i+2] == 1 && B[i-6] == 1 && B[i-12] == 1 &&
      B[i+5] == 1) {
         value=value+1
      }
   }
   If (i==15 || i==16) {     //Tiger blocked in spot B15 or 16
      If(B[i] == 0 && B[i-1] == 1 && B[i-2] == 1 && B[i+1]
      == 1 && B[i+2] == 1 && B[i-6] == 1 && B[i-12] == 1
      && B[i+5] == 1) {
         value=value+1
      }
   }
   If (i==17) {                  //Tiger blocked in spot B17
      If(B[i] == 0 && B[i-1] == 1 && B[i+1] == 1 &&
      B[i-2] == 1 && B[i-6] == 1 && B[i-12] == 1 &&
      B[i+5] == 1) {
         value=value+1
      }
   }
   If (i==18) {                  //Tiger blocked in spot B18
      If(B[i] == 0 && B[i-1] == 1 && B[i-2] == 1 && B[i-6]
      == 1 && B[i-12] == 1) {
         value=value+1
      }
   }
```

```
    If(i==19) {                          //Tiger blocked in spot B19
        If(B[i] == 0 && B[i+1] == 1 && B[i+2] == 1 &&
        B[i-5] == 1 && B[i-11] == 1) {
          value=value+1
        }
    }
    If (i==20) {                         //Tiger blocked in spot B20
        If(B[i] == 0 && B[i-1] == 1 && B[i-5] == 1 && B[i+1]
        == 1 && B[i+2] == 1 && B[i-11] == 1) {
          value=value+1
        }
    }
    If (i==21) {                         //Tiger blocked in spot B21
        If(B[i] == 0 && B[i-1] == 1 && B[i-5] == 1 && B[i+1]
        == 1 && B[i-2] == 1 && B[i-11] == 1) {
          value=value+1
        }
    }
    If (i==22) {                         //Tiger blocked in spot B22
        If(B[i] == 0 && B[i-1] == 1 && B[i-2] == 1 && B[i-5]
        == 1 && B[i-11] == 1) {
          value=value+1
        }
    }
}
```

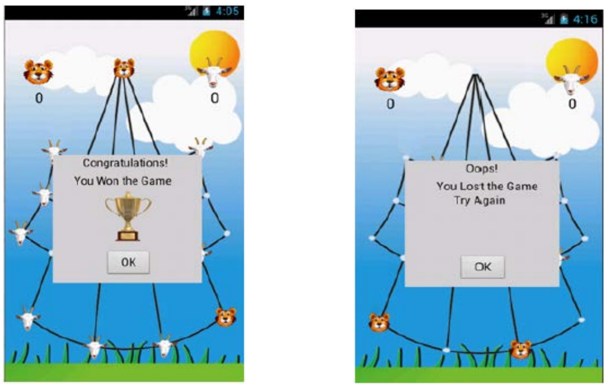

Figure 5. Result: (a) Human player won (b) Human player lost

*B. Playing for Defensive*

Whenever the Android detects an empty spot which is followed by T, G it may insert T in the empty spot or capture the G to prevent the blocking from G.

*C. Moves for Winning*

First, it is obvious that the goats have to consider the border during the placement phase - any goat that strays into the center will get eaten, or cause the demise of some other goat, without any apparent benefit in return for the sacrifice. Therefore, goat-player strategy sounds simple: first populate the borders, and when at full strength, try to advance in unbroken formation, in the hope of suffocating the tigers.

Second, tiger is moving back and forth, until near the end of the placement phase. Their goal is to stay far apart from each other, for two reasons: (1) in order to probe the full length of the goats' front line for gaps, (2) so as to make it hard for the goats to immobilize all the three tigers at the same time.

*D. Capture the Goats*

For the tigers, the objective is to 'capture' ten goats to win. Capturing is performed by jumping over the goats, although capturing is not obligatory. There are possible cases to capture the goat.

Algorithm:

Let B[i] be the ith spot. So I lies between 0 and 22. Let numerically T be represented by 0 and G by 1 and empty spot by 2.

```
If (i==0 && B[i]==0) {
    If(B[i+2] == 1 && B[i+8] == 2) {
          B[i]=2, B[i+2] =2, B[i+8] =0
    }
    If (B[i+3] == 1 && B[i+9] == 2) {
          B[i]=2, B[i+3] =2, B[i+9] =0
    }
    If (B[i+4] ==1 && B[i+10] ==2) {
          B[i]=2, B[i+4] =2, B[i+10] =0
    }
    If (B[i+5] == 1 && B[i+11] == 2) {
          B[i]=2, B[i+5] =2, B[i+11] =0
    }
}
If (i==1 || i==2 && B[i]==0) {
    If(B[i+1] == 1 && B[i+2] == 2) {
          B[i]=2, B[i+1] =2, B[i+2] =0
    }
    If (B[i+6] == 1 && B[i+12] == 2) {
          B[i]=2, B[i+6] =2, B[i+12] =0
    }
}
If (i==3 || i==4 && B[i]==0) {
    If(B[i+1] == 1 && B[i+2] == 2) {
          B[i]=2, B[i+1] =2, B[i+2] =0
       }
       If (B[i+6] == 1 && B[i+12] == 2) {
          B[i]=2, B[i+6] =2, B[i+12] =0
       }
       If (B[i-1] == 1 && B[i-2] == 2) {
          B[i]=2, B[i-1] =2, B[i-2] =0
       }
  }
  If (i==5 || i==6 && B[i]==0) {
       If(B[i-1] == 1 && B[i-2] == 2) {
          B[i]=2, B[i-1] =2, B[i-2] =0
       }
       If (B[i+6] == 1 && B[i+12] == 2) {
          B[i]=2, B[i+6] =2, B[i+12] =0
       }
  }
  If (i==7 && B[i]==0) {
       If(B[i+1] == 1 && B[i+2] == 2) {
          B[i]=2, B[i+1] =2, B[i+2] =0
       }
  }
  If (i==8 && B[i]==0) {
       If(B[i+1] == 1 && B[i+2] == 2) {
          B[i]=2, B[i+1] =2, B[i+2] =0
       }
       If (B[i+6] == 1 && B[i+11] == 2) {
          B[i]=2, B[i+6] =2, B[i+11] =0
       }
       If (B[i-6] == 1 && B[i-i] == 2) {
          B[i]=2, B[i-6] =2, B[i-i] =0
```

```
        }
    }
    If (i==9 || i==10 && B[i]==0) {
        If(B[i+1] == 1 && B[i+2] == 2) {
          B[i]=2, B[i+1] =2, B[i+2] =0
        }
        If(B[i-1] == 1 && B[i-2] == 2) {
          B[i]=2, B[i-1] =2, B[i-2] =0
        }
        If (B[i+6] == 1 && B[i+11] == 2) {
          B[i]=2, B[i+6] =2, B[i+11] =0
        }
        If (B[i-6] == 1 && B[i-i] == 2) {
          B[i]=2, B[i-6] =2, B[i-i] =0
        }
     }
     If (i==11 && B[i]==0) {
        If(B[i-1] == 1 && B[i-2] == 2) {
          B[i]=2, B[i+1] =2, B[i+2] =0
        }
        If (B[i+6] == 1 && B[i+11] == 2) {
          B[i]=2, B[i+6] =2, B[i+11] =0
        }
        If (B[i-6] == 1 && B[i-i] == 2) {
          B[i]=2, B[i-6] =2, B[i-i] =0
        }
     }
     If (i==12 && B[i]==0) {
        If(B[i-1] == 1 && B[i-2] == 2) {
          B[i]=2, B[i-1] =2, B[i-2] =0
        }
     }
     If (i==13 || i==14 && B[i]==0) {
        If(B[i+1] == 1 && B[i+2] == 2) {
          B[i]=2,   B[i+1] =2, B[i+2] =0
        }
        If (B[i-6] == 1 && B[i-12] == 2) {
          B[i]=2, B[i-6] =2, B[i-12] =0
        }
     }
    If (i==15 || i==16 && B[i]==0) {
        If(B[i+1] == 1 && B[i+2] == 2) {
          B[i]=2, B[i+1] =2, B[i+2] =0
        }
        If (B[i-6] == 1 && B[i-12] == 2) {
          B[i]=2, B[i-6] =2, B[i-12] =0
        }
        If (B[i-1] == 1 && B[i-2] == 2) {
          B[i]=2, B[i-1] =2, B[i-2] =0
        }
    }
    If (i==17 || i==18 && B[i]==0) {
        If(B[i-1] == 1 && B[i-2] == 2) {
          B[i]=2, B[i-1] =2, B[i-2] =0
        }
        If (B[i-6] == 1 && B[i-12] == 2) {
          B[i]=2, B[i-6] =2, B[i-12] =0
        }
    }
    If (i==19 || i==20 && B[i]==0) {
        If(B[i+1] == 1 && B[i+2] == 2) {
          B[i]=2, B[i+1] =2, B[i+2] =0
        }
        If (B[i-5] == 1 && B[i-11] == 2) {
          B[i]=2, B[i-5] =2, B[i-11] =0
        }
    }
    If (i==21 || i==22 && B[i]==0) {
        If(B[i-1] == 1 && B[i-2] == 2) {
          B[i]=2, B[i-1] =2, B[i-2] =0
        }
        If (B[i-5] == 1 && B[i-11] == 2) {
          B[i]=2, B[i-5] =2, B[i-11] =0
        }
    }
```

## IV. Conclusion

I expect this paper will grab the attention of a much wider body of traditional game researchers and that it will inspire others to play this fascinating game. Certainly, with time, new updates and modifications to the algorithm shall definitely improve the gameplay. But as we have seen, that fuzzy logic and making decisions based on multiple conditional cases has definitely a strong advantage over hard computing techniques and thus, soft computing is widely used in implementing the functionalities of various game engines used worldwide.

## References

[1] **"**Lambs and Tigers (Game).", Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc., date last updated (8 June 2016)
[2] **"**Android version history.", Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc., date last updated (14 June 2016)
[3] Mario Zechner and Robert Green, "Beginning Android 4 Games Development", New York, 2011
[4] Lim Yew Jin And Jurg Nievergelt, "Tigers and Goats is a draw", MSRI Publications, Volume 56, 2009
[5] Mark L. Murphy, "Android Programming Tutorials", 3rd edition, USA, 2010.
[6] Y. J. Lim and J. Nievergelt, Computing Tigers and Goats, ICGA Journal 27:3, 131–141, Sep 2004.
[7] J. Nievergelt,Exhaustive search, combinatorial optimization and enumeration: Exploring the potential of raw computing power, pp. 18–35 in Sofsem 2000: Theory and Practice of Informatics, edited by V. Hlava c, K.G. Jeffery and J. Wiedermann, Lecture Notes in Computer Science 1963, Springer, Berlin, 2000.
[8] Y. J. Lim,Using biased two-population co-evolution to evolve heuris tic game players for Tigers and Goats, Unpublished manuscript.
[9] R. Gasser, Solving Nine Men's Morris, pp. 101–113 in Games of No Chance, edited by Richard Nowakowski, MSRI Publications 29, Cambridge University Press, New York, 1996

## AUTHORS

I am Shouthiri Partheepan was born in Batticaloa, Sri Lanka, in 1988. I received the B.Sc Hons. Degree in Computer Science from the Eastern University, Sri Lanka, in 2015, and reading M.Sc in Computer Science from the University of Peradeniya, Sri Lanka.

In 2014, I joined the Department of Mathematics, Eastern University, Sri Lanka as a Temporary Demonstrator, and in 2015 became as an Assistant Lecturer. My current research interests include Mobile Applications, Artificial Intelligence and Image Processing.