© [2005] IEEE. Reprinted, with permission, from Agbinya, Johnson and Henney, Adnre 2005, 'Board Games of African Origin and Mobile Phones', 2nd International Conference on Mobile Technology, Applications and Systems, 2005, pp. 1-8.This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of the University of Technology, Sydney's products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to pubs-permissions@ieee.org. By choosing to view this document, you agree to all provisions of the copyright laws protecting it.

Board Games of African Origin on Mobile Phones

Andre J. Henney and Johnson I. Agbinya+++
Department of Computer Science, University of the Western Cape, Private Bag X17, Bellville 7535, South Africa
ahenney@uwc.ac.za
++Faculty of Engineering, University of Technology, Sydney; Bldg 1, Level 24, Broadway Campus Sydney, Australia
agbinya@eng.uts.edu.au

ABSTRACT

Most African board games have high state space complexity and limit their implementation on mobile phones and hardly any are. In this paper, we discuss the state space of such board games and investigate several African board games from the Mancala family. The paper provides a basis and rational for software games that match the broader interests from the continental context. We implement a version of the board games for the Nokia GPRS mobile phones and describe how the game was developed, and tested. The game application is for the Nokia 6310i mobile phone. We describe how the game was developed, implemented and tested on a mobile phone.

Index Terms- African games, Java, Mancala, Mobile Phone, OTA, PDA

I. INTRODUCTION

The games industry has become a new area for revenue generation for cellular network operators and is set to rival what is being achieved using ring tones. The paper covers an investigation into software implementation of games primarily played on the African continent with broader scope internationally. The motivation for such games stems from the fact that most games that are currently available on cellular handsets do not address the needs of the African market as very few of the Africans cellular phone users identify with them. On the African continent there is however a vast spread of diverse games that are being played by young and old. Due to this diversity the paper focuses only on African board games of the Mancala family. In what follows we address their game tree complexity and state space. The words Mankala and Mancala mean the same thing in this paper.

Most African board games have predominant use of even rows, holes and seeds. The following characteristics of the board games therefore involve even numbers:

- Rows of holes in the board games, usually two or four rows per board (except Gebeta from Eritrea)
- The number of holes per hole, usually, 2, 4, 6 or 8
- The number of seeds sown per hole at start of each game, usually 2 or 4 seeds per hole
- The number of players at a time is usually 2

Game Complexity

The complexity of a board game depends on the number of rows, holes and seeds sown at start of the game. It provides an assessment of the degree of difficulty involved to play the game and to convert its rules into software format. Four views on the complexity of games can be expressed as:

- State – space complexity
- Game – tree complexity
- Mutational complexity and
- Computational complexity of the algorithms used in its software production

Given the game tree and current state space, the best possible move can be found using the minimax algorithm. The alpha-beta pruning algorithm is typically used for this.

State-Space Complexity

The state-space complexity of a game is the number of legal positions that are reachable from the initial position of the game. Viewed as a two-dimensional pattern, we may also describe the state-space of a board game as the various pattern of distribution of seeds on the board, from initiation to the end of the game. The current state space can be used as an input to an evaluation function that is used to evolve a game. The output of the evaluation function can also be used in a mini-max search.

The state-space and game-tree complexities of several African board games were addressed by Wernham [1]. The state-space complexity of a board game is defined here as the various ways (permutations) available for setting up the game at start, the number of moves to establish winning result, the variation in the number of illegal moves and complex moves. The expression defining state-space complexity combines the number of holes used in the game, the number of seeds involved and the number of illegal or improbable positions. The state-space complexity C(h, s, k) of African board games is:

$$C(h,s,k)=\frac{(h-1+s)!}{(h-1)!s!}-k \qquad (1)$$

Where:

h = number of holes
s = number of seeds
k = number of illegal/improbable positions

For example, a board game with no illegal moves (k=0), 12 holes and 48 seeds has state-space complexity of 7.2381×10^{70}. If there are improbable positions in the game, the state-space complexity is reduced by that number. An improbable position in

Oware would include 47 seeds in one hole, and the remaining seed in another.

The state-space complexities of several African board games are compared in Table 1. The initial set up complexity is unity for games like Oware (Awari) and Bao (Zanzibar) with unique seed contents in each hole.

In Omweso (Uganda), each player has 32 seeds to set up, giving 7.5 x 10^{11} possible positions for the first player to set up, which can be countered by the other player in 7.5 x 10^{11} ways giving 5.6 x 10^{23} combinations [1]. As noted in [3], "in tournament play there are no illegal set-up combinations, and very few improbable ones.

The 'k' factor for mankala games is lower than in positional games since many pieces may share the same 'hole' in mankala. "State-space complexity in Omweso rises rapidly after the first captures, and remains high throughout as the seeds become redistributed in large numbers quasi-stochastically from one player to the other".

Game	Initial set	As play starts	Midgame	Endgame
Awari	1	Slowly rising	2.8 x 10^{11}	Falling
Bao	1	Very slowly Rising	10^{25}	-
Omweso	5.6 x 10^{23}	Rising quickly	10^{25}	-

Table 1: State-space complexities of 'world' games – Adapted from [1]

There are implicit assumptions leading to the values in Table 1, therefore the values are approximations. Different games have different numbers of seeds in play at different stages. For example, in Bao re-entrancy is permitted and seeds are introduced throughout the initial phase of the game. Captured seeds are re-entered onto the board.

Games with high theoretical state-space complexity may be less 'intricate' for humans as the outcomes are beyond mental calculation and require 'brute force' calculations of no finesse [2]. This implies that the interesting games are games that are on the edge of human capacity for calculation and tactics and especially those that show high degrees of chaotic behaviour. This observation was initially made in [2].

Game Tree Complexity

Game tree complexity is the mathematical term used to describe a directed graph that illustrates the number of possible positions in the game. The complexity may be shown in terms of a tree (a diagram) that shows the manner in which the game progresses for every move on the board. A large directed graph reflects a great game-tree-complexity. The complexity is an indication of the number of states in the game.

Game-tree-complexity may be defined as the product of the game's average branching factor and the number of half-moves (plies). Each board game forms a tree that evolves as the game is being played. This concept should be understood under the general semantics of data networks. A tree is formed when a series of operations lead to others and those steps can be represented pictorially as logical growth of the tree from one node to the next.

Game tree complexity is the number of leaf nodes in the solution search tree of the initial position of the game. By implication, games that have more than one initial position also have several game tree complexities. Game tree complexity provides an estimate of the number of branches in the tree at set-up used to represent the game based on a number of players, the branches per move and the length of the play. Game-tree complexity can be calculated as:

$$G_t = i_1 \times i_2 \times b^p \qquad (2)$$

Where,

i = branches in set-up of game for players 1 and 2

b = branches per move

p = plays in game length (average game length)

The search space is given by the number of branches raised to the power of the average length of the game.

The game tree complexity of Oware, Bao and Omweso are given in Table 2 [1]. The complexity values are affected by the state-space at initial set up. For example, Omweso has a large variation in the state-space at start up and leads to a large value of endgame permutations.

Game	Initial set	By endgame (no forced moves)
Awari	1	2.8 x 10^{32}
Bao	1	2 x 10^{34}
Omweso	5.6 x 10^{23}	(5.6 x 10^{23}) (5x 10^{50}) = 2.8 x 10^{74}

Table 2: Game-tree complexities of African board games

Another expression that can be used to estimate the game-tree complexity is given as the product:

$$G_t = \prod_{k=1}^{d} w(k) \qquad (3)$$

where w is the branching factor, and d the average length of the game. In this expression, every step of the game is modelled as starting from a node with w branching factors or w different options available to the player. With d=57 and w=4 the game-tree complexity for Bao is given by the following:

Game	depth	Complexity (G_t)
random-play		2.8 x 10^{34}
random evaluation	4	3.1 x 10^{37}
random evaluation	10	1.5 x 10^{37}
fixed evaluation	4	5.0 x 10^{28}
fixed evaluation	10	5.7 x 10^{28}

Table 3: Game-Tree Complexity of Bao [3]

In Oware, feeding of the opponent's holes when they are all empty is permitted. This may be considered as a forced move. Omweso has nothing of that nature. However, in Bao forced moves are very common [1], perhaps 1/10 in master-level plays

and more common amongst less experienced players who do not know how to avoid or take advantage of these situations.

In Bao a player has choices of where to re-enter captured seeds (left or right) during the initial stage of the game. In Omweso however, the player has choice of where to begin sowing to make a capture but not where to re-enter the seeds. These considerations must be made in estimating the game-tree complexity.

Mutational Complexity

Like in genetic mutations, the state spaces of games mutate. This concept was first pinpointed by De Voogt [2]. The mutational complexity of a game is the number of changes on the board due to a single move. Moves change the nature of the board by altering the state-space, the positions and the number of seeds in the holes due to planted new seeds.
In Bao capturing re-entry rules genetically change the nature of the front row as new seeds are planted there. Mutational complexity is less pronounced in Oware since there is no relay sowing. Therefore the impact of a move is limited to a few holes. In what follows, we first describe a typical Mancala board game of 12 holes – 2 on either side. Then we simplify the level of the board game to match small form factor devices and limit the number of rows and columns to 2 and 3 respectively. Each hole may also contain only 2 seeds.
The paper covers an investigation into software implementation of games primarily played on the African continent. The motivation for such games, stem from the fact that most games that are currently available on cellular handsets are mostly from non-African origin. On the African continent there is a vast majority of diverse games that are being played by young and old. Due to this diversity the project focuses only on African board games. A survey of the Mancala family of games was done. The project yielded a game called Henney's Mancala, based on the game Mini-Mancala.

The game was implemented and developed using the Java programming language. Implementation of the game was done for the Nokia 6310i [9]. The final project allows any General Packet Radio Switched (GPRS) enabled cell phone user to download a game from a games server using one of the two methods that exist for mobile game deployment, namely direct cable connection or OTA (Over the Air), using the software package Sun ONE Studio Mobile Edition [10], Java 2 Micro Edition (J2ME) from Sun Microsystems. The project also enlists the use of software package from Nokia Group

II. Background

Recently the IEEE reported that the game of Oware has been solved [4]. Oware is a member of the Mancala family of board games. "Mancala is the term to denominate games with one shared characteristic: moves are not executed as in chess or checkers, instead moves are executed by sowing seeds (or other playing pieces) into holes. Mancala games occur mainly in Africa and Asia, and in parts of the New World settled by natives of those regions" [5].
Mancala, also referred to as the hole-seed-sow-capture game, derived from the components of the game which consists of a game board with holes carved into it of identical shapes and play pieces. The play pieces can by anything from seeds to pebbles. In this paper we call the play pieces seeds. The game is played by sowing seeds in order as described below. Almost all Mancala board games are played in similar style with slight variations as the game played in different African and international settings.
Almost all Mancala board games are played in similar style with slight variations as the game played in different countries or in Africa.

III. Survey of Mancala Games

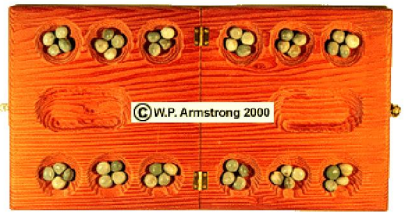

Figure 1: A Wari board from the Caribbean island of Grand Cayman with 4 seeds in each hole **[6]**

The games summarized in this document are all part of the Mancala family of games. The object of the Mancala games is to capture more seeds than your opponent. The game board consists of 14 holes, two holes are used to hold the captured seeds (these holes are not part of the playing board). The other 12 are situated on the playing board in two rows of 6. Each player owns a row of six holes.

Basic rules of Mancala Games

To start the game the 48 holes are divided equally to fill the 12 holes, each hole will then contain 4 stones. Player's alternate turns playing and must make move on each turn. "Playing" or making a move is when a player picks up all the stones in any of the six holes of his row and then in an anti-clockwise movement starts depositing a stone in to every hole following the hole from which he/she started.

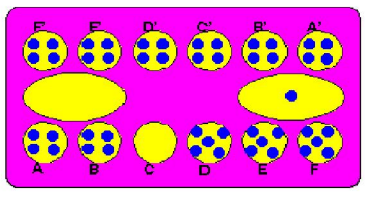

Figure 2: In this board 4 seeds from hole C are picked up and deposited one each in holes D, E, F and own home store [6].

If the player has a balance of seeds after deposits in his/her holes, it must be deposited in the opponent's holes, until all the seeds are used up.

No holes may be skipped when depositing the seeds in any of the holes. If however a player has enough seeds to end up at the starting hole (such a move is called a kroo[1]), then only must that hole be skipped and the following hole may be used.

A player may capture seeds from the opponent only if his/her last deposit is in an opponent's hole and, the amount of seeds after the deposit are either two or three.
When a player finds that he/she has no more seeds to move, which means all his/her holes are empty then that player moves all the remaining seeds on the board to his/her capturing hole as shown by figures 3 and 4. The player with most seeds then wins the game. When a player who is about to move sees that the opponent's holes are empty he/she must, if possible, deposit seeds into the opponent's row.
The game can also end up in a form of "stale mate", whereby there are so little seeds on the board that the player's merely end up moving the seeds around the board. If no capture is possible, the players may decide to end the game and then just count the captured seeds without counting the ones that are left. The player with the most seeds wins the games.

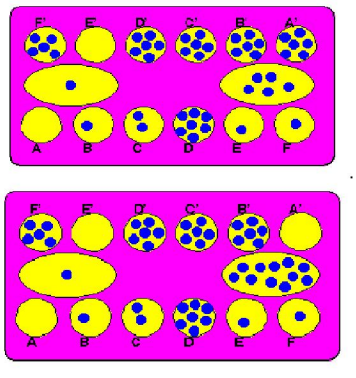

Figure 3 & 4: Since your last seed landed in your empty hole F, you take all the seeds in your opponent's hole A' and add them to your home store [6]:

We based our game on MiniMancala which is a shortened version of the Mancala family of games.

A. MiniMancala

1. *The game components*:

The game consists of board of 2 rows, 2 pits and, 8 seeds. Figure 1 shows the game board for MiniMancala.

[1] A kroo is when a player has enough seeds to go all the way around the board.

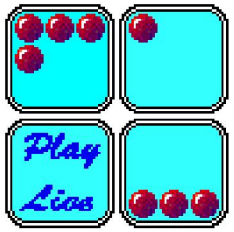

Figure 5: MiniMancala Game Board [8]

2. *Rules of MiniMancala*

- To start the game the 8 seeds are divided equally to fill the 2 pits, each pit will then contain 2 pieces.
- There are two players, north and south with each player controlling the two pits on his/her side.
- Player's alternate turns playing and must make move on each turn.
- "Playing" or making a move is when a player picks up all the seeds in any of the 2 holes in his row and then in an anti-clockwise movement starts depositing a seed in to every of the three holes.
- If there is more than three seeds in a pit then the player continues depositing the seeds in the pits until he/she has no more.

A player wins the game when he/she has captured all the seeds in his row at the end of his/her move.
Our developed game differs from MiniMancala in the sense that we added two more holes and the number of seeds per hole can not be more than 6.

IV. GAME LOGIC

A two dimensional integer array was used to represent the game board in memory. Integer values in each cell of the array represented the seeds. This was very convenient, because the representation on the screen of the mobile phone would then be the same. As in the original Mancala games seeds is sown in an anti-clockwise rotation.

2	2	2
2	2	2

Figure 6: Initial array representation of board game

Each time a play (sow) is made, each of the holes adjacent to the hole from which the sow was initiated is increased by 1. That hole is then set to 0.

3	3	0
2	2	2

Figure 7: Array representation of game after a sow

When a play is made and seeds have to be sown in the opponent's holes, then the array acts as if it is a circular array. It however does keep track of the two dimensions therefore it would never seem as it has to little array positions.

3	4	1
2	2	0

Figure 8: Array representation after opponent's sow

4	0	1
3	3	1

Figure 9: Array representation after player a sow. Shows how the seeds are sown in circular motion.

After each sow the program checks the total amount of hole in each player's holes and if it is less than 2 then that specific player looses the game.
Each hole may not contain more than 6 seeds therefore the program also needs to check this after each sow and then might needs to skip a hole with 6 seeds and continue to sow after that hole. The basic rules set for Mancala games have been retained with some extra rules added.

V. IMPLEMENTATION

The game MiniMancala was chosen for this project and adapted to include two more holes and one or two extra rules. This game was chosen because it conformed to the required characteristics for the project which are:

- Size
- Difficulty
- Rules
- Easy to implement

The following section describes the game development process in an abstract manner.

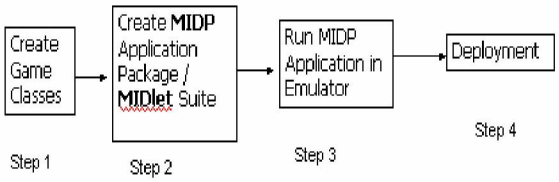

Figure 10: Software development steps

1) Step 1

The game MIDP (Mobile Information Device Profile) classes were developed at this stage.
The classes include:

- Initial_Screen
- Options_Screen
- Game_Screen
- WinOrLoose_Screen
- TopScore_Screen
- InstructionsHelp_Screen
- About_Screen

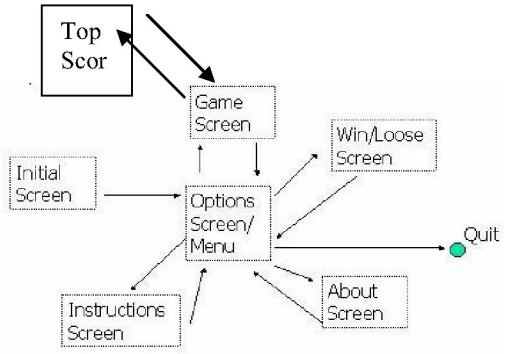

Figure 11: Screen transitions at the start and during game play

The above figure gives a broad overview of the transition from the start of the game until the end of the game. Clearly the user will firstly encounter the initial screen, which will just display the name of the game.
Secondly the options screen/menu screen will display a list of items/choices to the user. The list will lead into any of the other screen depending on what option the user takes. The menu screen would be our main screen, because all the other screens should be able to lead back to this screen (hence the back arrow). The user should also be able to quit from the menu screen.
Depending on the user choice any of the other screens would then appear on the screen. All the screens will be in the form of classes.

2) Step 2

The creation of the MIDP Application package or MIDlet suite is automatically done by Sun One Studio. This process is the creation of two files called the Jar and Jad files.
Jar file – is the file that contains the classes and other MIDP files needed to run the game on a mobile phone.
Jad file – describes the MIDlet and the contents of the Jar file.

3) Step 3

When the game is developed and ready for testing, any of Nokia's emulators can be installed in order to test the game. This can be integrated with Sun One Studio and Nokia's Developers Suite. The emulator allows the developer to test the functionality of the game as if it is being tested on an actual Nokia mobile phone. The Nokia 6310i emulator is being used for this project. This part of the project would not require any deep analysis, because it will be done using the software.

4) Step 4

There are 2 ways in which the game can be deployed to a mobile phone:

1. Device Deployment – The game can be deployed via IrDA (Infrared connection) or Serial cable connection using the Nokia PC Suite.

Figure 11 explains that the deployment can be done from the computer on which Sun One Studio Mobile Edition is installed. This option was the most viable for this project. Another method that the user can employ is to download the JAD file to its personal computer via the Internet from a webpage. The JAD file can then be deployed via any of the two methods above.

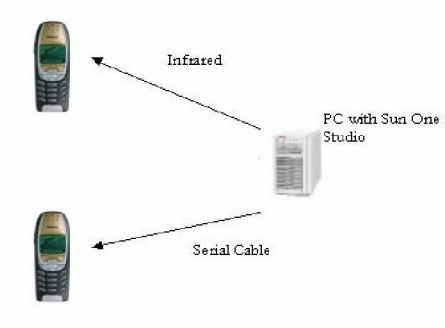

Figure 12: Device deployment

2. Server Deployment - Figure 12 explains how the Computer running Sun One Studio opens a PassiveFTP connection with a MIDlet Server (situated at a Service Provider) and then transfers the files needed to run the game to this server. The game is then downloaded from the MIDlet sever to the mobile phone via WAP or GPRS.

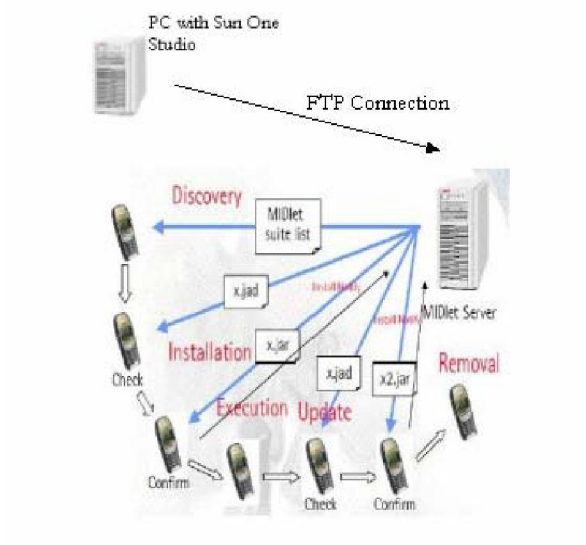

Figure 13: Server deployment

All classes were implemented in the Java programming language. The Sun One Studio 4 Mobile Edition IDE in conjunction with Nokia's Developers Suite for J2ME (User Guide - Appendix D) were used to develop and deploy the chosen game.

VI. SCREENSHOTS

Figure 14: Screenshots of game running on Nokia 6310i mobile phone.

VII. DATARUNS

The following shortened data runs were printed to the monitor screen for testing purposes when the game was executed using the emulator. It shows the initial stage as well as how the game board and array will change after play.

2 numSeeds 2
Cell and Number of Seeds in hole

2 3 0
2 2 2
Graphic representation of board for testing purposes

Player 1 Played
3 3 0
2 2 2
Player 1 Played
PC cols4
Changing of board after each sow by player

1 numSeeds 2
3 3 0
2 0 3
Computer Played
3 3 1
2 0 3
Computer Played

Changing of board after each sow by computer

VIII. USER'S GUIDE

1) Rules of the game

- Henney's Mankala is a one player game. The mobile/cellular phone user can play the game against a computer user.
- Each player (user and computer player) has 3 cells/blocks to play in.
- Each cell will be filled with two (2) seeds at the start of play. Each player cannot accumulate more than seven (7) seeds in a hole.
- The user/human player has to start the game by sowing the seeds in a particular hole of his/or her choice. Seeds are then sown in an anti-clockwise direction.
- The user/human player can select from the options menu to play a easier game, by selecting to switch ON, the level option from the Options menu.
- By default, any player that accumulates two (2) or less seeds in his/her three (3) holes wins the game.
- When the user chooses to play the easy level, the player who then accumulates three (3) or less seeds in his/her three holes, wins the game.
- The player will not be able to sow from an empty hole. When the computer player encounters an empty hole it will automatically skip that hole and start sowing from a hole that contains at least one (1) seed.
- A player will also not be able to sow in a hole containing seven (7) seeds; automatically seeds will be sowed in the first allowable hole.

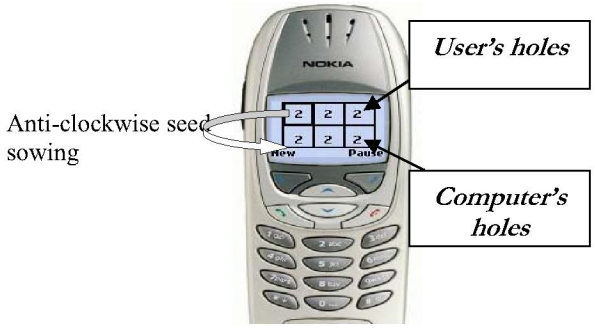

Figure 15: Actual game display

2) Playing the game

- The user can choose any option from the menu, that will be displayed on-screen. The menu options are as follows:
  - New
  - TopScore
  - Options
  - Instructions
  - About
  - Exit
  - Continue(only displays after the game has been played/paused)

Figure 16: Menu screen

- If the user chooses the New option, then he/she will be taken to the game screen to play the game.
- The user must then select a hole from the top row of holes to start playing(the bottom row belongs to the computer player). In order to select a hole the user must use keys 4 (left) and 5 (right) to move the active box to the appropriate hole.
- In order to sow the seeds the user will have to press key 5 (Fire) to sow the seed(s).
- The computer will automatically play after the user has played, this is instantaneous.
- If the user pauses the game to change any options, then the continue option will be added to the menu to allow the user to continue playing his/her game.

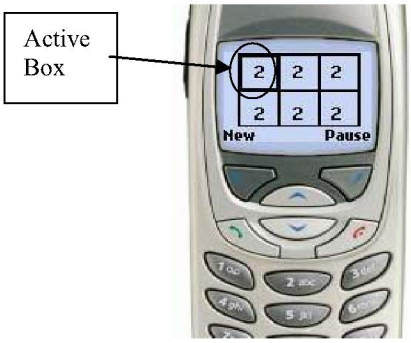

Figure 17: Game screen

3) Choosing other menu options

When the user chooses any of the following options:

- New & Continue
  - Discussed in previous section.
- About
  - A brief description of the author of this game.
- Instructions
  - A description on the rules of the game.
- TopScore

- Shows the scores of the player, if more than one game is played in succession.

- Options
  - Allows the user to choose from two extra game options namely, vibrate and level. By default these options are turned OFF. Here they can be turned ON or OFF.
  - Vibrate – Allows the user to feel the Nokia 6310i vibrate during game play.
  - Level – allows the user to play an easier game.
- Exit
  - Allows the user quit playing the game.

4) Phone setup

Key	Option/Command
4	Left
5	Execute
6	Right
Soft Key Left	Select
Soft Key Right	Back / Pause
Soft Key Scroll Up	Scroll Menu Up
Soft Key Scroll Down	Scroll Menu Down

Figure 18: Key mapping to command

Figure 19: Actual key mapping on mobile phone

5) Caveats

- The game does not allow the user to see whether the computer has finished playing. Computer playing is instantaneous.
- The user is not allowed to play (sow) from an empty hole. The developer of this game assumes that the user will abide by the rules of the game. Therefore there is no checking whether a user/human player plays in an empty hole.

IX. Conclusion

Mobile phones have become the dominant form of wireless communication in the world, and therefore are suitable medium for distributing African games. The game developed here represents not only relevant but also culturally satisfying content for the African mobile phone market.

X. Acknowledgment

Andre thanks Professor J.I. Agbinya for his supervision of this project. We acknowledge the Department of Computer Science, University of the Western Cape for their support.

XI. References

[1] Brian Wernham, "Omweso: The Royal Mankala Game of Uganda – A General Overview of Current Research", International Omweso Society, http://www.omweso.org, 2002, pp. 1-26.

[2] De Voogt, "Limits of the mind", Leiden, CNWS, 1995

[3] Jeroen Donkers and Jos Uiterwijk, Programming Bao, IKAT, Dept. of Computer Science, Universiteit Maastricht, Maastricht, The Netherlands, 2003

[4] Visa Korkiakoski and Janne Pänkälä, Warri (adaptation of mancala), http://www.hut.fi/~vkorkiak/mancala/doc/html_rules/

[5] http://www.gamecabinet.com/rules/BaoIntro.htm .

[6] http://waynesword.palomar.edu/ww0603.htm

[7] Ancient Africa http://www.cmi.k12.il.us/Urbana/projects/AncientCiv/africa.html

[8] http://www.mindsports.net/CompleteGames/Other/MiniMancala.html

[9] Nokia Tools and SDK's http://www.forum.nokia.com/main/1,6566,030,00.html

[10] Sun ONE Studio Mobile Edition http://wwws.sun.com/software/sundev/index.html