Proceeding of the International Conference on Advanced Science, Engineering and Information Technology 2011

Hotel Equatorial Bangi-Putrajaya, Malaysia, 14 - 15 January 2011

ISBN 978-983-42366-4-9

Game Development for Smart Phones Based on Local Heritage

Oras F. Baker[1], Kasthuri a/p Subaramaniam[2], Gayathri A/P Megeswaran[3], Aloysius Akpanobong[4]

UCSI University

Faculty of Management and Information Technology

Kuala Lumpur, Malaysia

[1]*orasbaker@ucsi.edu.my*

[2]*kasthurisubr@ucsi.edu.my*

[3]*gayathri@ucsi.edu.my*

[4]*aloyakpa@ucsi.edu.my*

***Abstract*— The market for mobile games is expanding rapidly, and several games market in the advanced world are booming with sales, that has led us to the use of latest tools for mobile application development to modify and enhance an existing and traditional game to make it available for mobile users, Congkak game is developed for the latest mobile phones in the market. This version has been modified to an eight holes, one store, and one player game for a start. Each player hole contains four seeds, which are sown continuously until the seeds are exhausted. The main objectives of the game are: to make it available for mobile phones as well as extend/modify the games' functionality.**

Major development tools used were the Sony Ericsson KToolBar, Netbeans IDE, Scite text editor, and Adobe Photoshop. UML class diagrams have also been modeled for the classes. The programming language used in this game development is java, programming in the micro edition platform (J2ME).

.

***Keywords*— Mobile Computing, Game development.**

I. INTRODUCTION

Congkak game is a Malay game (from Malaysia) and Indonesian traditional variant of 'mankala', a large group of related games that are played almost all over the world.

Mankala games are also known as 'pebble-and-pit games' or 'count-and-capture games' and are played on a board that contains 2, 3 or 4 rows of holes [11].

Traditionally, these games are played as a 2-player game by simply digging holes in the soil (usually sixteen holes in total, seven holes containing seeds and one additional hole as a store for each player). One player and three player versions are also available.

In 1960, Kalah, a first computerized version of the game, was produced and many others followed. Mankala games are played with a large set of equal counters (Irving et al, 2003). These counters can be pebbles, shells, seeds or any small round objects. The game starts with a certain distribution of an equal number of the counters over the pits. A move is made by selecting one of the holes, emptying all counters out of it and putting back the counters one-by-one in adjacent holes in an anticlockwise direction. This is called 'sowing'. The hole in which the last counter is put determines what happens next. Sometimes a capture takes place and the turn is over, sometimes the sowing continues, and other times the player is allowed to do another move. The goal of the game is always to capture as many counters as possible.

There is a variety of board sizes for mankala games as well as many variations in the rules. Some of the games are very easy to play while others are extremely difficult to master. Murray (1952) and Russ (2000) group the mancala games together by the number of rows on the board and also by some specific rules. In the group played in South East Asia, a capture is allowed if the last counter is put in an empty hole on the player's side. Congkak belongs to this group.

The aim of this paper is to describe a congkak version developed for mobile phones using the java programming language and J2ME. The game play has been described in detail. The objectives of the game have also been highlighted, and the choice of java as the programming language used has been justified. Furthermore, the design and implementation phases have been discussed, outlining

the software and hardware tools used; the development process, and the list of functions. The last part contains discussions and indication of future research and functionalities to be added.

II. GAME OBJECTIVES

The market for mobile games is expanding rapidly, and many markets in the developed world are booming with sales. This offers a compelling need for developing compelling and engaging games [1]. Mancala is a very popular game and there already exist various traditional and computer based versions of this game. A documented version of a mancala-type game for mobile phones is the "Mancala: FS5" released January 28, 2009 by FlipSide5 and sells for US$1.99 and runs only on iPhones and iPods [2].
The main objectives of developing this game are to:

- Extend/modify the functionality of the conkak traditional game and make it suitable for deployment on mobile phones.

- Develop a mobile-game version of the traditional Malay conkak game.

III. APPLICATION DESIGN

This game is designed using the java programming language, programming in the micro edition platform (J2ME), while the SonyEricsson KToolBar is used to test run the application. J2ME is a java platform designed to specifically target consumer devices and electronic appliances, including wireless devices such as cell phones and Palm PDAs [4]. The many benefits of J2ME include:

- Extends the original "Write Once, Run Anywhere" design philosophy to the wireless world. Wireless applications developed using Java can run on different devices from different vendors, thus greatly improving the program's portability [4].
- Saves development time and cost, considerably improving productivity [4]. This factor is particularly critical in today's fast-paced, highly competitive market.
- Designed with network capabilities in mind, providing a rich set of network libraries that makes writing network programs much easier [5].
- Provides a dynamic deployment mechanism that allows applications to be downloaded and installed onto devices over the wireless network, thus also allowing users to download applications on demand and personalize their on-device applications dynamically [5].

To make it easier for future enhancements, object oriented programming was used. Functionalities have been broken into various classes/modules. In all, there are ten classes to handle the different functionalities.

Figure 1: mobile mankala running on SonyEricsson KToolBar Emulator

IV. IMPLEMENTATION

The images used in this game were created using Paint.Net. Paint .Net was chosen because of its support for transparent png files [6]. The images include the hand, board, seeds, and holes (big and small).

V. STARTING THE GAME

The class that runs the application is the *CongkakMIDlet* class. Here, a new Congkak instance (canvas) is only instantiated if the canvas is null. This avoids running multiple instances at the same time, thus saving memory. Also, in implementing the commandAction method, destroyApplication( ) is made false when the user wants to exit, and the notifyDestroyed( ) is called so that the application exits properly. This is most useful in situations where the user is in the middle of a game and suddenly wants to exit. By calling the notifyDestroyed( ) method, the instance of that game is properly terminated [7].

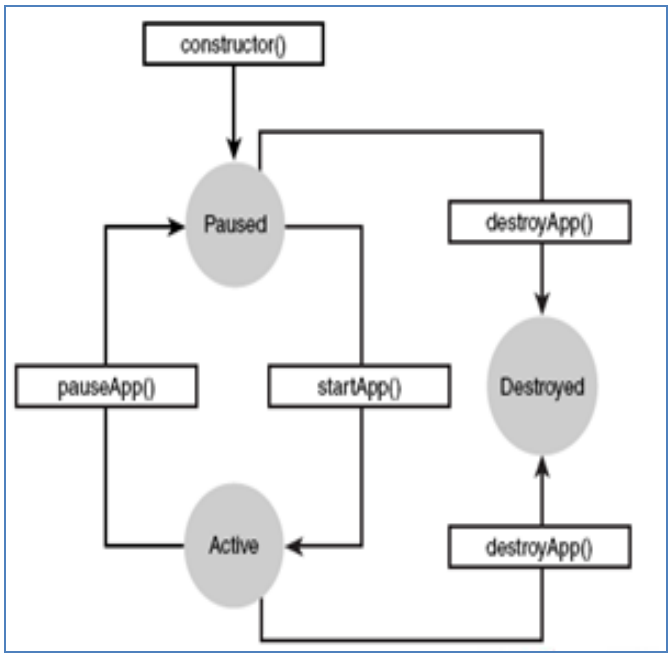

Figure 2: MIDlet life cycle-the three abstract methods startApp(), pauseApp(), and destroyApp() must be implemented.

The Background class, as the name implies, creates the background (non-animated).

VI. Rendering To Screen

The class used to render to screen and provide extra handling to user inputs (key pressed states) is the javax.microedition.lcdui.game.GameCanvas [8]. The GameCanvas super class provides two major advantages over the Canvas super class and these are:

- the application has control over exactly when the display is updated, instead of having to wait for paint() to be called by the system software [9].
- The programmer can control which section of the screen needs to be updated [9].

```
//Rename paint to draw:
public void draw(Graphics g){
//Do the painting just like before...
}

//Graphics loop:
public void run(){
    Graphics g = getGraphics(); //Get the
graphics
    try{
        while(true){
            draw(g); //Call the new paint
method
            flushGraphics(); //Make the
screen update
            Thread.sleep(10); //Sleep between
updates
        }
    }catch(InterruptedException e){
    }
}
```

Figure 3: Code snippet for GameCanvas example

This Congkak class extends the GameCanvas super class, and so it becomes possible to override many of its abstract methods in order to implement custom rendering [10]. In the gameInitialize() method, the variables bgWidth and bgHeight have been used to set the screen resolution to 240 and 320 respectively. This means that this game will work perfectly with mobile phones of this resolution or screen size. The fullScreenMode(true) method is called to set it to full screen mode. updating the game state takes into account all the actual processing of the game logic, such as moving the hand, changing the number of seeds in the holes, emptying the seeds in the adjacent hole when the last seed in the players' hand end up in an empty hole, and checking the number of seeds in the big hole to set game win or game over. All these game updates are actually handled in the individual classes that manipulate the hand and seeds respectively (HandManager and CongkakSeeds).

The startCanvass method handles the filling up of seeds in the holes as well as updating the seed frames. The seed image was drawn as a picture frame, measuring 440 *120 pixels. This image was further subdivided into 11 rows and 3 columns, each measuring 40*30 pixels. The first frame was left blank, and the second frame starts with 1 seed. Each additional frame has a +1 seed increment, up to the thirty third frame which contain 32 seeds, since the maximum number of seeds any hole can hold is 32.

The getKeystates() method from the Java API was used to listen to key states. Since this game player needs only to move the hand in one direction, only the UP_PRESSED key state has been used therefore all other keys are not working for now.

VII. Checking For Game Over State

Checking for game over state is a necessity for all games [11]. This takes place either when the player loses the game or beats it, so it's either a lose or a win state.

In this game, the player loses the game if the life counter becomes zero, and the number of seeds in the store is less than 20. He wins if the number of seeds in the store is greater than 20, regardless whether he still has lives or not. A conditional if statement block inside the startCanvas() method is used to check for this state.

VIII. FUTURE WORK

This game is just the first draft to see if it can be implemented. There is need to add more enhancements in order for improvement. The getWidth() and getHeight() methods need to be used in order to overcome the limitation of running it perfectly on only hand phones with 240*320 pixels resolution [5]. These methods are implemented in the java API to get the screen resolution of the platform that the application is running on.

A list also needs to be added as the first screen to allow users select from a menu. The menu list would contain items like start, pause, exit, continue, rules, and about, for example. Codes will then be implemented to handle these actions individually.

There is also the possibility of making it more challenging, like playing against the hand phone. For this however, an artificial intelligence (AI) algorithm need to be implemented so that the hand phone can respond accordingly to user actions.

Finally, it could also be networked so that two players can play against each other, either in the same location (in which case blue tooth connection can be enhanced) or in different locations (where advance networking could be implemented).

IX. CONCLUSION

In this paper, we discussed the reasons for developing this game, and the choice of java as the programming language.

We also discussed the implementation behind it and how some of the codes work.
Finally, we talked about the future enhancements that can be added to this game to make it more challenging and exciting.
This game holds great potentials if improved, considering the popularity of the original mankala board game that it is derived from.

REFERENCES

[1] Hall et al, A Lightweight Rule-Based AI Engine for Mobile Games, 2004, Retrieved from http://osiris.sunderland.ac.uk/~cs0lha/Publications/2004/mimosa-ace.pdf, On July 19, 2009

[2] Sellers, D. FlipSide5 releases Mancala: FS5 board game for the iPhone (Macsimum News: Jan 28, 2009). Retrieved from http://www.macsimumnews.com/index.php/archive/flipside5_releases_mancala_fs5_board_game_for_the_iphone/, On July 20, 2009

[3] Kushchu, I. Positive Contributions of Mobile Phones to Society, 2007, Retrieved from http://www.mgovernment.org/resurces/positivezfinal.pdf, On July 20, 2009.

[4] Programming Wireless Devices with the Java2 Platform, Retrieved from http://book.javanb.com/Programming-Wireless-Devices-with-the-Java2-Platform/0321197984_ch02lev1sec3.html, On July 20, 2009

[5] Feng, Y & Zhu, J, Wireless Java Programming with J2ME, Sams Publishing, Indianapolis, 2002, 10-11.

[6] [Rowlingson, P. Paint.Net 2.0. Retrieved from http://www.v3.co.uk/vnunet/downloads/2129101/paint-net, On July 19, 2009

[7] Giguere, E, Understanding J2ME Application Models, 2002, Retrieved from http://developers.sun.com/mobility/midp/articles/models/, On July 18, 2009.

[8] Wells, MJ, J2ME Game Programming, Thomson Course Technology, Boston, 2004, 601.

[9] Li, S and Knudson, J. 2005. Beginning J2ME, From Novice to Professional. Apress, 255.

[10] Bornander, F. 2009. Mobile Game Programming for Beginners part 1 of 4.Retrieved from http://www.codeproject.com/KB/mobile/J2MEForBeginners1of4.aspx?display=Print, On July 19, 2009.

[11] Irving, G, Donkers, J. and Uiterwijk, J. Solving Kalah. 2000. In: ICGA Journal 2000; 23 (3).11, 1997.