NOWADAYS MOBILE SOFTWARE. AN EXAMPLE

Dan-Costin Tuşaliu
Adrian-Gabriel Neaţu
Cosmin Selaru
Manole Ecaterina

University of Craiova

Abstract: A mobile game is a video game played on a mobile phone, smartphone, PDA, handheld computer or any type of handheld or wireless device. Mobile games are played using the technologies present on the device itself. For networked games, there are various technologies in common use. Examples include text message (SMS), multimedia message (MMS) or GPRS location identification. More common, however, are non networked applications that simply use the device platform to run the game software. The games may be installed over the air, they may be side loaded onto the handset with a cable, or they may be embedded on the handheld devices by the OEM or by the mobile operator. This paper is focused in the development of a golf game using J2ME technology.

Keywords: Java, Mobile Software, Game, Engine, Interface, Pixel, Graphics

1. INTRODUCTION

Mobile games are developed using platforms and technologies such as Windows Mobile, Palm OS, Symbian OS, Macromedia's Flash Lite, DoCoMo's DoJa, Sun's J2ME (Java 2 Micro Edition, recently rebranded simply "Java ME"), Qualcomm's BREW (Binary Runtime Environment for Wireless), WIPI or Infusio's ExEn (Execution Environment). Other platforms are also available, but not as common.

Mobile games tend to be small in scope and often rely on good gameplay over flashy graphics, due to the lack of processing power of the client devices. One major problem for developers and publishers of mobile games is describing a game in such detail that it gives the customer enough information to make a purchasing decision. Currently, Mobile Games are mainly sold through Network Carriers / Operators portals and this means there are only a few lines of text and perhaps a screenshot of the game to excite the customer. Two strategies are followed by developers and publishers to combat this lack of purchasing information, firstly there is a reliance on powerful brands and licenses that impart a suggestion of quality to the game such as Tomb Raider or Colin McRae and secondly there is the use of well known and established play patterns (game play mechanics that are instantly recognisable) such as Tetris, Space Invaders or Poker. Both these strategies are used to decrease the perceived level of risk that the customer feels when choosing a game to download from the carrier’s deck.

Recent innovations in mobile games include Singleplayer, Multiplayer and 3D graphics. Virtual love games belong to both of singleplayer and multiplayer games. Multiplayer games are quickly finding an audience, as developers take advantage of the ability to play against other people, a natural extension of the mobile phone’s connectivity. With the recent internet gambling boom various companies are taking advantage of the mobile market to attract customers, Ongame the founders of PokerRoom developed in 2005 a working mobile version of its poker software available in both play money and real money. The player can play the game in a single player or multiplayer mode for real or play money.

As well, the MMORPG boom seems to hit the world of mobile games. According to their website CipSoft has developed the first MMORPG for mobile phones, called TibiaME. SmartCell Technology, a mobile applications developer, is in development of the first cross-platform MMORPG called Shadow of Legend. Shadow of Legend will have the ability to play on both a PC and a mobile device.

The dominant mobile software platform is Java (in its incarnation as "J2ME" / "Java ME" / "Java 2 Micro Edition"). J2ME runs atop a Virtual Machine (called the KVM) which allows reasonable, but not complete, access to the functionality of the underlying phone. The JSR process serves to incrementally increase the functionality that can be made available to J2ME, while also providing Carriers and OEMs the ability to prevent access, or limit access to provisioned software.

This extra layer of software provides a solid barrier of protection which seeks to limit damage from erroneous or malicious software. It also allows Java software to move freely between different types of phone (and other mobile device) containing radically different electronic components, without modification. The price that is paid is a modest decrease in the potential speed of the game and the inability to utilise the entire functionality of a phone (as Java software can only do what this middle-man layer supports.)

Because of this extra security and compatibility, it is usually a quite simple process to write and distribute Java mobile applications (including games) to a wide range of phones. Usually all that is needed is a freely available JDK (Java Development Kit) for creating Java software itself, the accompanying Java ME tools (known as the Java Wireless Toolkit) for packaging and testing mobile software, and space on a web server (web site) to host the resulting application once it is ready for public release.

2. SPECIFICATION

This engine offers support to the differences of level that determine the acceleration or slowing down of the ball. To vary at maxim the game possibilities, the texture of the playground will not be uniform, containing variation of level, water zones of teleportation that will alter the track of the ball depending on the situation.

In structure, the game contains 18 different playgrounds with different difficulties ranges, which can be played in one of the three different modes:
- progressive mode: one only player can play any playground from the 18 in one of the following configurations: any three of them, any six of them, the first nine of them or all 18;
- two players: the two participants will play on the same playground and will execute successively hits depending on the chosen configuration from the three;
- practice mode: it is chosen one of the playgrounds for practice and in this way the statistics of the game are not modified.

3. OPTIONS AND INTERFACE ELEMENTS

When entering the playground, depending on the cross over option, it is realized a motion for pre-visualizing the playground. In any moment of the game, the user can move the perspective by pressing the numerical buttons, to analyze the whole playground.

The physics of these motions over the playground is very realistic, using elaborated data structures to represent the playground, and takes into consideration the different types of texture the playground has: grass, water, concrete. Also, when editing the levels and in the representation of the code, there ware taken into consideration the level sensing of the playground, the absorption effects for the ball and the obstacles in the game. The power of the hit is established each time by the selection of a bar with a non linear variation that oscillates between two extremes.

Playing a playground begins by selecting a start position, It is permitted, in this way, moving in a more restricted area before making the first hit, to realize a better positioning regarding the direction of the hit.

Once the start position has been chosen, the direction of the hit is chosen by the circular moving of an indicator dotted-line. This moving is accelerated or decelerated to shorten the needed time for choosing the desired direction. The direction line has also a helping role, its dimension varying according to the obstacles met in the given direction.

After choosing the direction, the strength of the hit is chosen. A varying scale is used in this matter. The oscillation varies continuously from the minimum value to the maximum value. By pressing a selection button the certain strength is chosen according the positioning of the cursor on the scale.

In this way the ball moves on the playground also according the texture of the zone that can automatically influence the speed and direction of the ball, also with the absorbing holes.
Another situation is when the ball reaches outside the playing area or in no-collision areas, as water holes. In this situation, the ball is automatically set at the beginning of the playground. There is a maximum of 10 hits on each playground, and when this limit is reached the play on that playground is ended.

3.1 The menu system

The game has an easy usable interface that contains a main menu and a pause menu, each one with some submenus.

In the main menu there will be accessible the following options:
CONTINUARE: permits to continue the game in a progressive manner from the point where it was left or saved.
JOC NOU: realizes the beginning of a completely new game (from the first playground) in the progressive mode.
MOD DE JOC: permits the selection of one of the three game types available.
STATISTICI: offers statistical information about the evolution in the progressive mode such as: numbers of holes played, number of playgrounds finished with one hit, the best total score for all the playgrounds, the total number of hits under a certain value called "par".
OPTIUNI: contains a set of game options such as activate/deactivate the music, activate/deactivate the game sounds, the vibrations, the perspective motion. Also, to vary the facilities of the game it is permitted for the player to choose between the progressive mode or practice mode. The last option in the menu is to reset the game, which means to erase all information related to status of the game, as well as configuration of the game, and options of the game coming back to their default values.
AJUTOR: presents information about the two sub-options: CONTROL and MOD DE JOC;
DESPRE: contains general information about the project.

The tree structure of the main menu is graphically represented in Fig. 1.

The Pause menu offers the following functionalities:

REIA JOC: realizes the return in the game;
TABELA DE SCOR: contains the PAR value for each playground together with the score obtained by the player;
REJOACA TRASEU: replay playground; permits restarting the current playing playground (without resetting the number of hits);
TERMINA JOC: finishes the current session of the game and returns to the main menu;
AJUTOR: it's the same option offered by the main menu;
OPTIUNI: contains the same set of options with the exception of the choosing the personage, because this cannot be changed during the game.

The Pause menu is presented in Fig. 2.

Fig. 1. Main menu. It presents the tree structure of the main menu.

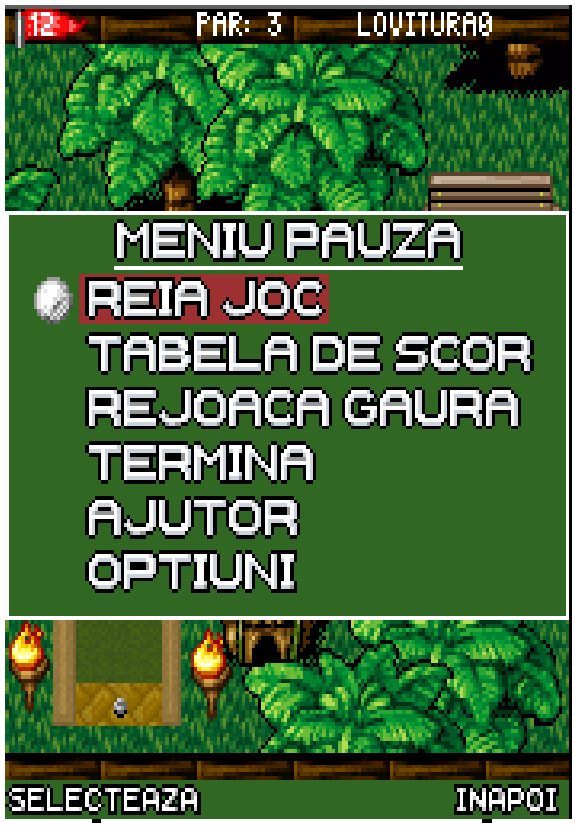

Fig. 2. Pause menu. Presents the tree structure of the pause menu.

3.2 The structure and representation of the playgrounds

Each playground has the dimension of 256x256 pixels and possesses a certain structure represented in the binary files from where the information is taken to generate them. This minimal structure has the role to diminish the information retained for each level. Because the dimension of the JAR archive has to be as small as possible, the resources must use as less space as possible. This is the reason for which there are used tiles and sprites to generate all the game levels.

For each playground from the 18 available there exists a bin file that offers all necessary information to generate a playground for a level. These files have been generated with the aid of a level editor that will not be presented here.

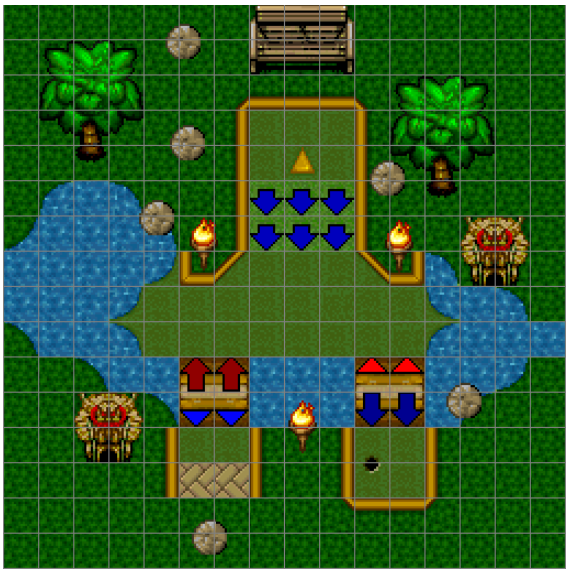

Fig. 3. Playground. The view of a playground made of tiles and sprites.

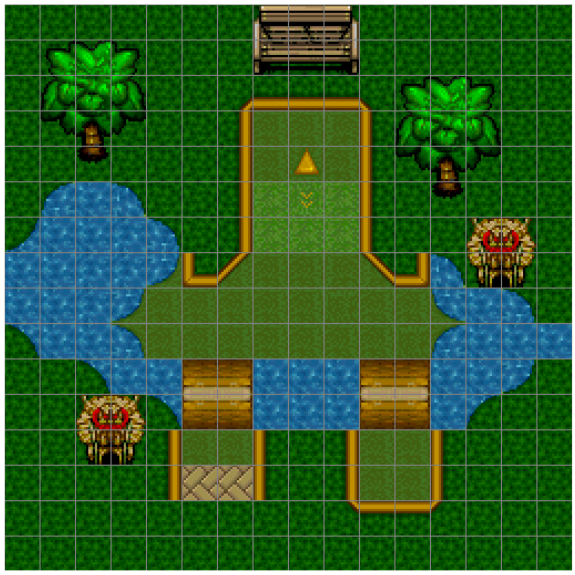

Fig. 4. Tiles level. Is the elementary texture and graphics level .

In Fig. 3 there is represented a playground in the edit state. It is structured on three levels:
1- the 16x16 tiles matrix that contains the playground view, as in Fig. 3
2 – a sprites set having the role to enrich the view with different objects and animations , as in Fig. 6.
3 – a set of field modifiers that changes the call direction giving it a certain acceleration (simulates the angle effect), as in Fig. 5.

4. THE CODE ANSAMBLE STRUCTURE

The code is structured in two principal classes and one interface:

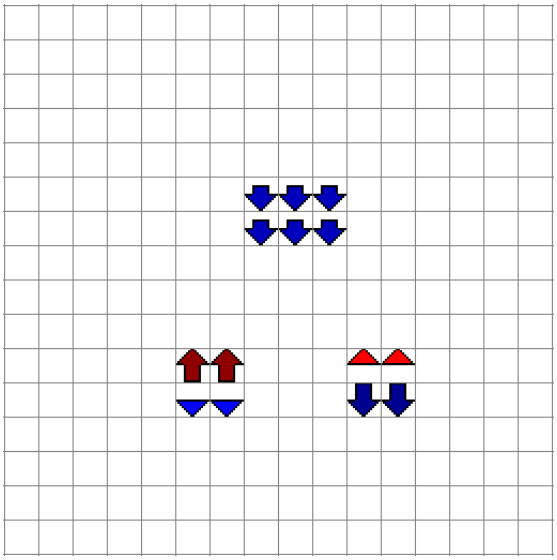

Fig. 5. Angle level. Shows the direction of the ball on the playground.

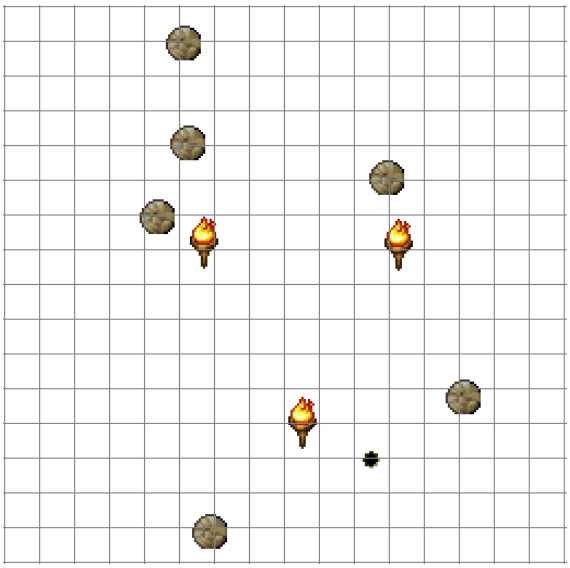

Fig. 6. Sprites level - the ornamental role in the view.

globals interface: contains the declarations for the majority of the global variables of the program;

- **MG** class: is the MIDlet class of the application; this implements the life cycle methods of the game and creates the engine thread in the startApp() method.
- **engine** : contains the working motor of the game. It extends the *Canvas* class to have a direct access for the graphical context of the apparatus. It implements the **globals** interface from which it takes all the global variables, and also the *Runnable* interface, and it will function in a separate thread startedb the startApp() method in the MIDlet class.

CommandListener interface is implemented to benefit the command facilities,, even if the game functions in **fullScreen** mode. This is made by using the joint part for the events of pressing the commands by the listner and drawing on the screen the corresponding software buttons.

PlayerListener interface is another listener that is used to intercept the audio events with the role to start the background music when a game sound end its play mode, by the playerUpdate() method.

The MG class code is presented here that sets the application course in the interaction with the Application Sistemul de Gestiune al Aplicatiilor :

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;

//clasa care extinde MIDlet
public class MG extends MIDlet
{
 private static engine Engine=null;

 public MG()
 {

 }

 protected void destroyApp(boolean flag) throws MIDletStateChangeException
 {
     Display.getDisplay(this).setCurrent(null);
    //opreste engine-ul
     engine.bIsRunning = false;
    //notifica AMS ca s-a efectuat terminarea MIDlet-ului
    notifyDestroyed();
 }

 protected void resumeApp()
 {
     //notifica iesirea din pauza
    Engine.breakExit=true;
     //reia functionarea
    Engine.Resume();
 }

 protected void pauseApp()
 {
    if(Engine!=null)
    {
        //reseteaza variabilele de tastatura (ultima tasta apasata, starea sa)
                Engine.cachedKeyCode = 0;
                Engine.nKeyState= 0;
                Engine.clearKeys();

    }
   //notifica AMS ca MIDlet-ul a intrat in pauza
   notifyPaused();
 }

 protected void startApp() throws MIDletStateChangeException //TS [12/4/2005] redesigned
 {
     //la prima intrare este creat si pornit engine-ul
    if(Engine==null)
    {
                try
                {
                        Engine=new engine();
                        Engine.startme(this);
                }
                catch (Exception e)
                {
                        System.out.println(e);
                }
    }
     //se revine din pauza
    else
    {
                Engine.breakExit=true;
                Engine.clearKeys();

    }
  }

}
```

5. CONCLUSIONS

The games for mobile equipment, though they have many constraints, represent a manner to enrich the creativity in development. Also, the usage of consumed resources using J2ME it is much lower than in PC games.
The development cycles are short, just couple of months usually, offering the possibility to realize different games in the same time and increases the possibility to create a greater success.

The possibilities for further development in this application refer to:

- realizing different variants that permits multiplayer gaming over Bluetooth transmission or even WAN connection

- realizing a 3D version in the moment when the technology will permit the transition of the 3D graphics API on the mobile phones.

The distribution on larger areas of mobile phones that support Java has created a market for this type of games, better than that for PCs, and with much lower costs that make them accessible for everyone. So their popularity increases and overcomes the PC games that are still better in graphic and other facilities.
Once realized, a success in the mobile phones games can be continued with adding new levels or facilities, representing a continuous attraction for a longer period of time on the market.
This project is an example of such a game that offers a small piece of entertainment for the moments that the recreation with a mobile phone is more in hand. By the offered facilities can be also an interactive entertainment, because it can be also played in two players' mode, successively.

A very attractive facility will be also playing with several other players simultaneously by Bluetooth. This will be possible only after the standardization of API for Bluetooth on more phones. In the moment this is found only on some phones and the specifications for the protocol are not yet standardized so they do not permit synchronization and communication between any phone models.

Once the computing power for the phones will grow, the 3D graphics API will be distributed and the interest for developing such variant will grow.

REFERENCES

Yuan, M. J., (2004) "Enterprise J2ME: Developing Mobile Java Applications", Prentice Hall

Knudsen, J., Li, S.,(2005) "Beginning J2ME: From Novice to Professional, Third Edition (Novice to Professional)", Apress,

Wells, M. J., Flynt, J. P., (2005) "Java ME Game Programming, 2E", Thomas

Hamer, C., (2005) "Creating Mobile Games: Using Java ME Platform to Put the Fun into Your Mobile Device and Cell Phone", Apress