

Attila Váradi (10030313)

Internet and Mobile Services

Free University of Bozen-Bolzano

01/28/2010

1. System functionalities	3
2. Human Computer Interaction.....	4
3. Code Structure:.....	8
4. Technical problems:	11
5. Future development:.....	11

1. System functionalities

General description:

OrienteeringGame is an orienteering mobile game. It was made to people who want to stay fit and tourists. Tourists can discover a city with this application.

The user gets various missions/tasks during the game that are usually to find a place or a building. User gets the target's coordinations and address. The essence of the game is that user must reach the target places or buildings in a given time. If he/she can't reach it, the game will end.

Functionalities:

- **Getting new random task:**

Tasks are given to the application as resource. When application requests a new task , it generates a new random number that is not previously. It guarantees that task doesn't repeat itself within a game.

- **Log function:**

The application saves the user's current coordinates when location is being updated to a RecordStore. Save also happens at every minutes. When the user completes a task or exit from the game the RecordStore is saved in a file on the storage drive. After this RecordStore is deleted.

- **Updating location:**

Location is updated at set intervals. This interval can be changed in the Setting menu.

- **Sending the log file to a servlet:**

When the user exits from the application the log file is sent to a servlet. This function is not complete yet. Application sends the log file, but the servlet can't save it. The aim of this function is that after the game the user's movements can be shown in a map.

2. Human Computer Interaction

When starting the application at the first time:



Figure 1. Choosing the storage drive

For the first time when application is starting the user must choose the storage drive as shown in Figure 1. It's necessary for saving the log file.

Starting a new game:

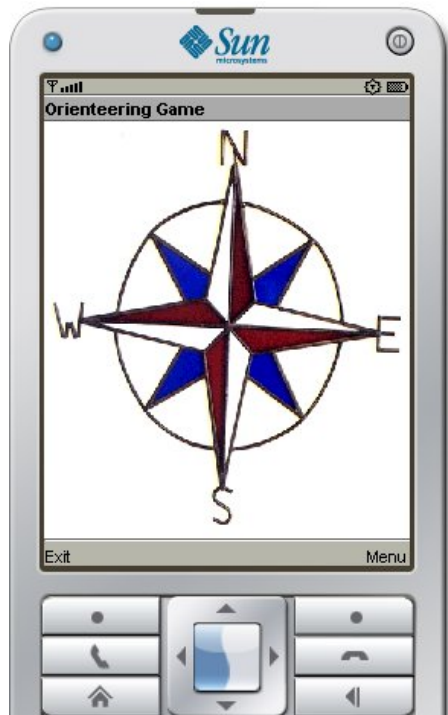


Figure 2. Main form

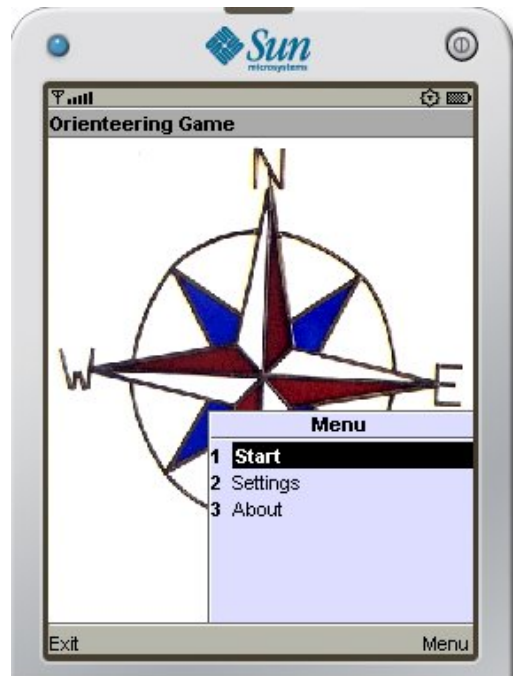


Figure 3. Options of the Main menu

The main form is very simple. It consists only one background image and a menu. Options of the Menu are shown in Fig. 3.



Figure 4. About

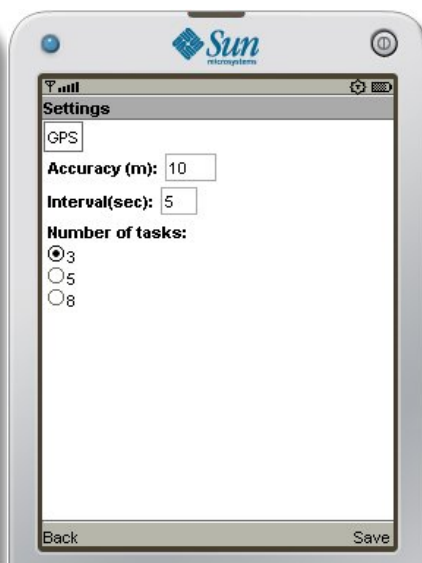


Figure 5. Setting form

Before the game user can set some options, such as number of tasks, accuracy of gps, interval of updating location.

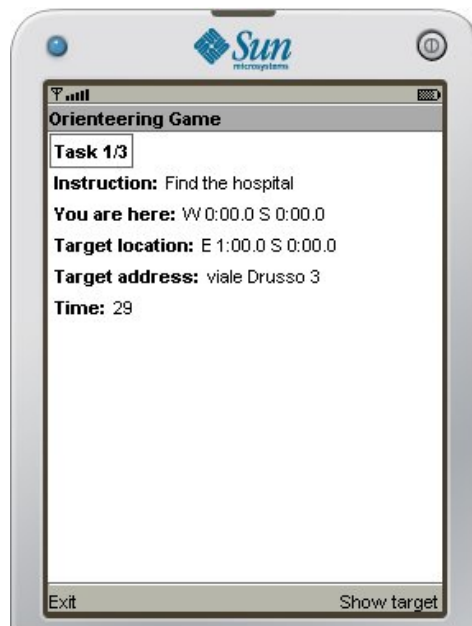


Figure 6. Game form

After starting the game the game form is appeared (see Figure 6.). Here, we can see the number of the current task, the instruction, the current coordinates, the target's coordinates, the target's address and the remained time. During the game tasks are chosen randomly.

By clicking the Show target, an arrow will be shown that shows the direction to the target and the distance between the current coordinates and the target's coordinates. The degree of the arrow is measured to the north. Arrow only shows the correct direction if the user is moving as shown Figure 7.

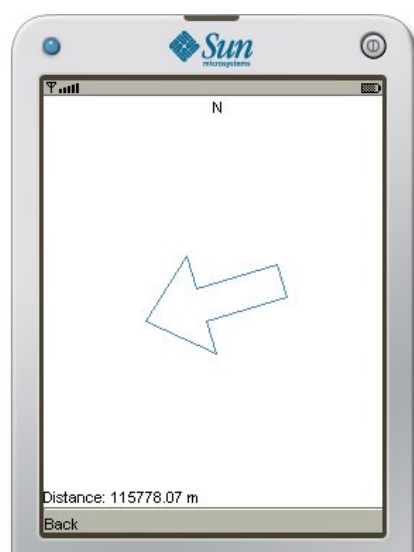


Figure 7. The helping arrow

End of the game:

If the user can not reach the target before the time is up, „Game over” alert will be shown.



Figure 8. Game over

If the user completes all task in the given time, „You won” alert will be shown with some data.



Figure 9. You won

3. Code Structure:

The application contains 7 classes, 1 MIDlet and 2 resource files.

Classes:

- **Setting:** This class contains the all necessary data. For example: coordinates, number of current task, degree between current position and the target's position... During updating location coordinates the most of these data is being updated.
- **AbstractShape:** This class describes a shape. This wasn't written by me. I used this class from „LocateMe” project.
- **Arrow:** This class implements the arrow. This wasn't written by me. I used this class from „LocateMe” project.
- **ArrowCanvas:** This is the canvas where the arrow and the distance are shown.
- **LogService:** This class implements methods which are necessary to log the user's actions and to save the log file.
- **DataStore:** The methods of this class get the randomly choosen tasks.
- **Point:** This class implements a point.

MIDlet:

- **OrienteeringGameMIDlet:** The main application logic is in this class. It implements CommandListener, LocationListener and Runnable.

Resource files:

- **Tasks.txt:** This file contains data about the missions/tasks.
- **Compass.png:** This is the background picture.

The following figures show the methods of each class and MIDlet:

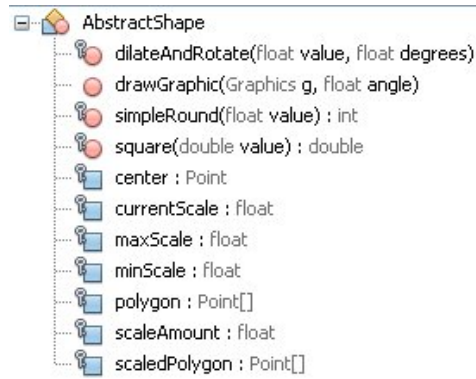


Figure 10. Class AbstractShape

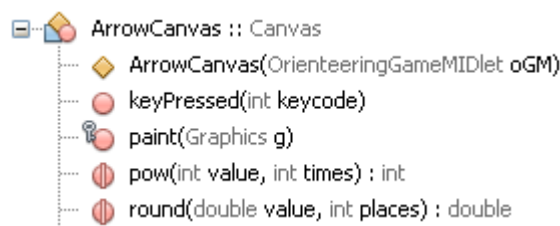


Figure 11. Class ArrowCanvas

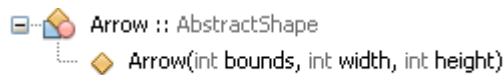


Figure 12. Class Arrow

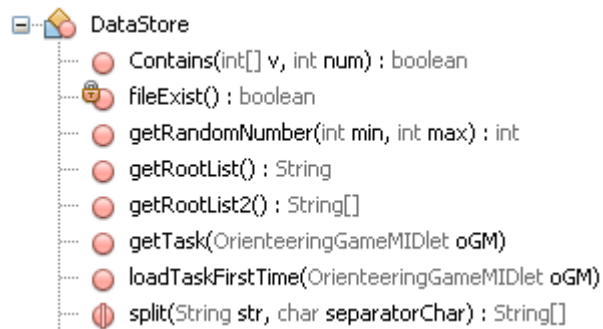


Figure 13. Class DataStore

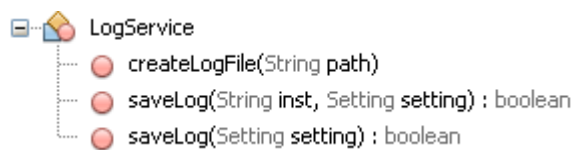


Figure 14. Class LogService



Figure 15. OrienteeringGameMIDlet

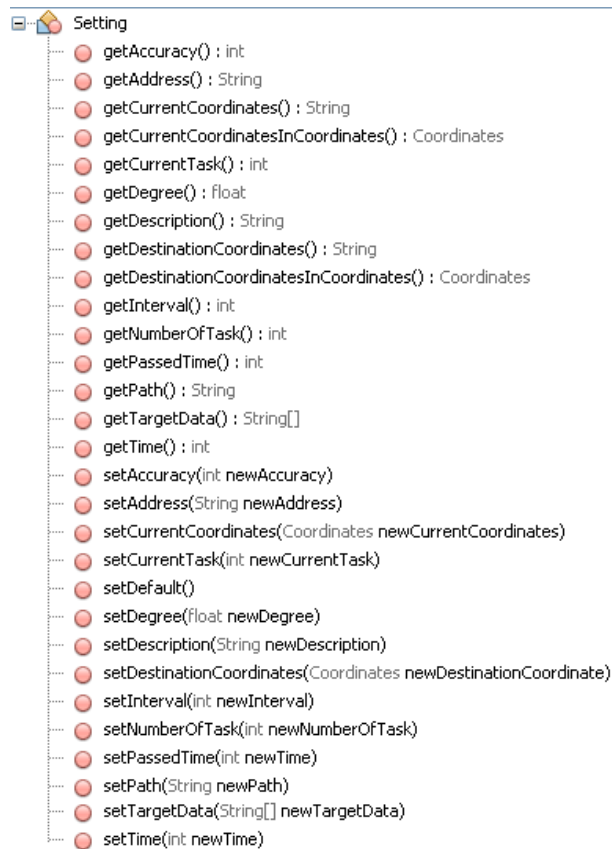


Figure 16. Class Setting

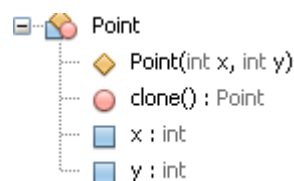


Figure 17. Class Point

About LocateMe project:

It's a location based mobile application under Common Public License v1.0 developed by Silent Software. I used some classes for my project. (Arrow, AbstractShape, Point).

you can download from:

http://mirror.viperfang.net/silentdevelopments/downloads/locateme/locateme_source_1.0.1_beta.zip

4. Technical problems and solutions:

- **Getting a new random task:** I had to generate a random number that wasn't so far. To do that I used an array in which I stored the random numbers. When I generated a new number I search it in the array. If there was a match I generated a new one and checked it again.
- **Updating location at set intervals:** I used a LocationListener that implements the LocationUpdate method to solve this problem.
- **Decreasing time by 1 minute:** For this I used a timer with a TimeTimerTask. In this TimeTimerTask, I decrease the time by 1 and save the log to the RecordStore. If the time is equal to 0, it means the user can't reach the target and the game will end.

5. Future development:

- Complete the „Sending the log file to a servlet” function.
- User can download other task data files to his/her mobile and the task is chosen from that files.