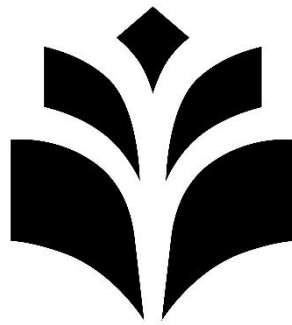


# Automatic Game Generation for O-Mopsi Mobile Orienteering Game

Anton Tsypchenko

Master's thesis



UNIVERSITY OF  
EASTERN FINLAND

School of Computing

Computer Science

June 2015

UNIVERSITY OF EASTERN FINLAND, Faculty of Science and Forestry, Joensuu  
School of Computing  
Computer Science

Automatic Game Generation for O-Mopsi Mobile Orienteering Game: Title of Thesis

Master's Thesis, 49 p., 1 appendix (2 p.)

Supervisors of the Master's Thesis: Professor Pasi Fränti

January 2012

**Abstract:** O-Mopsi is a location-based game which requires player to reach goals in the real world. The aim of the game is to visit all goals in a given game in free order as fast as possible. The game software is on all major smartphone platforms. Games to be played are carefully created by selecting suitable goals from a predefined collection. The game master looks for features as:

- reachability
- location source
- naming

In this thesis, we attempt to automatize the game creation process. It is also possible to use it as a draft for to manually created games. We describe the game generation process in detailed. How it works overall, we also describe heuristic algorithms such as greedy and Tabu search for solving travelling salesman problem. We introduce a definition what is a good game, and base on it, we implement a new feature to generate a good game that fits for the requested parameters.

**Keywords:** Location-based aware, orienteering, travelling salesman problem, mobile application.

## **Foreword**

This thesis was done at the School of Computing, University of Eastern Finland during the spring 2015.

I want to express my gratitude to my teachers from NKSU who have helped me attain a decent level of knowledge that allowed me to study as IMPIT student.

I would like to express my gratitude to my supervisor professor Pasi Fränti for the useful comments, remarks and engagement through the learning process of this master thesis. Furthermore I would like to thank Radu Mariescu-Istodor for introducing me to the topic as well for the support on the way, for editing my grammar mistakes and leading me towards.

I thank my family for supporting me emotionally and mentally, and financial when I got stuck or needed reclusion. Without the support, encouragement, and dedication to assist me, this master thesis would not have been possible.

## **List of abbreviations**

ACM	Association for Computing Machinery
UEF	University of Eastern Finland
GPS	Global Positioning System
TSP	Travelling salesman problem
O-Mopsi	Orienteering Mopsi
SQL	Structured query language

# Contents

1.	Introduction.....	1
1.1	Location-based games.....	2
1.2	Geocaching .....	2
1.3	Tourality.....	3
1.4	GPS Mission .....	4
2.	O-Mopsi .....	6
2.1	Game rules .....	6
2.2	How to play.....	7
2.3	O-Mopsi mobile software .....	9
3.	Creating a game .....	14
4.	Criteria for a good game .....	20
5.	Automatic game generation .....	27
5.1	Finding possible goals .....	29
5.2	Automatic goal ranking .....	30
5.3	Goal clustering.....	32
5.4	Evaluation of the quality of the game .....	34
5.5	Game complexity .....	35
5.6	Reference tour by Tabu search algorithm.....	38
5.7	Greedy algorithm for travelling salesman problem .....	42
6.	Experiments .....	43
7.	Future work improving quality of the goals .....	46
8.	Conclusions.....	47
	References.....	48

## Appendices

Appendix 1: Code Usage (2 pages)

# 1. Introduction

A game is structured playing, usually undertaken for enjoyment and sometimes used as an educational tool<sup>1</sup>. A video game is an electronic game that involves human interaction with a user interface to generate visual feedback on a video device[1]. Development of the video gaming industry attracts more people to the personal computers and game consoles. Taking them into a virtual world and reduces physical activity. Problem of the modern population and especially young people, have long worried about health care. Development of exergames allows player to move from virtual world to the real world. Player achieves good results in the game is closely associated with increased physical activity which helps player to improve his/her physical shape[2].

*Exergames* is a term using for video games with remote controls and motion sensors that require the players to move<sup>2</sup>. Konami's *Dance Dance Revolution* was cited as one of the first major successes of exergaming<sup>3</sup>. The game uses special PlayStation controller (Figure 1). Player has to use feet to push buttons in the same moment when the button appears on the display. Location-based games are a subset of exergames. This type of games is played in a pre-defined *scenario*. There is a prepared list of tasks, usually a list of the places that player must visit to finish the game. Some location-based games required physical access to the location from the game master to create a scenario for the game. For some games such as GPS mission, scenarios can be done remotely. Those games have got web server and the persons use computer tools to create a scenario. This person is called game master. Game master uses web-tools to create a scenario for the game at any place in the world. It requires time to create new scenarios. To save time of those people and to provide new scenarios to the players in real-time we need automated process for creating scenarios for the games.



Figure 1: Dance Dance Revolution controller

---

<sup>1</sup> <http://www.merriam-webster.com/dictionary/game>

<sup>2</sup> <https://www.acsm.org/docs/brochures/exergaming.pdf>

<sup>3</sup> <http://www.foxnews.com/story/2005/01/18/exercise-lose-weight-with-exergaming.html>

## 1.1 Location-based games

*Location-based* games (geo-games) are a type of games where player's physical location affects the game process. Games have rules such as reaching points in limited amount of time or repeating a tour. *Geo-games* use satellite navigation systems to obtain the location[3].

Game types can be classified as *online* and *offline* games, or whether they use *absolute* or *relative* location [4]. Online games require good internet connection to communicate with other players and get updates about game situation. Tourality game has multiplayer mode; user knows location of other players on the same a team member. Usually online games require exact location.

Offline games do not need internet connection to play. For games using absolute location player needs to download the game data to the device, and after that he/she is able to play anytime. Offline games do not support multiplayer possibility. When relative location is used, player must change location, keep the exact speed, or perform some other tasks based on relative change in position[5].

## 1.2 Geocaching

*Geocaching* is a traveling game using satellite navigation systems. The *Global Positioning System* (GPS) is a space-based satellite navigation system that provides location and time information in all weather conditions. When the device detects four or more GPS satellites it is possible to determine location of the GPS-receiver location[6]. The purpose is to find *caches* hidden by other players. The basic idea is that some community members hide caches and report their GPS coordinates through Internet. Other players use these coordinates and their GPS-receivers to find the caches<sup>4</sup>. They are usually stored in hermetic box or even camouflaged containers (Figure 2) [7].



Figure 2: Caches

At first glance it seems that the search for the cache is trivial using a GPS-receiver. However, the accuracy with which the device determines the position ranges from few to tens of meters. This provides only an approximation area of the cache location. To find the container player must use clues from the description of

---

<sup>4</sup> <https://www.geocaching.com/guide/default.aspx>

the cache, observe the environment, and rely on intuition and his previous experience.

To create the game, community members have to hide caches, save geographic coordinates from GPS receiver and post this data with the clues on how to find it. To automatically generate such game is not possible, because the cache is a physical object. Maybe in the future drones will be popular enough to use them to hide or relocate objects and perhaps take photos as clues to help players find them.

### 1.3 Tourality

*Tourality* is another location based game. Real environment such as an urban district, a park or even a forest serve as a virtual playground where players location and movements are identified via GPS-signal using smartphone. The objective is to reach predefined *spots* in the game by moving towards them with your mobile device<sup>5</sup>.

In the multiplayer mode player can compete against other gamers, join forces in a team or even compare their *highscores* with other gamers in the Tourality community. Unlike Geocaching, Tourality focuses on time: the aim is to be faster than the others. In doing so, it is players own choice whether you play the *game sets* by feet, use roller skates, a bobby car, or other motorized vehicles. You just need to be fast! Game uses *gold* as the currency. Game set is a number of geographical spots, set of rules and game type. Player can buy information when there is a hidden loot, power up to visit the next spot instantly, and power down (for enemies) to blind all players in the game for 2 minutes. Blind player cannot see the location of the spots and other players on the map. From each game, player accumulates scores, which are used for global leaderboard<sup>6</sup>. This motivates people to play more[8].

Tourality uses game sets with different playing modes: *race*, *trail*, *chase*, *rush*, *act*. Player is able to play single, versus and team versus modes<sup>7</sup>. Example of a trail game in London is shown in Figure 3. On the top of the screen there are game standings and time. Majority of the screen contains the map with the reached goals (green) and inactive goals (grey.) The orange stars are *Goodies* which may contain a certain amount of gold or hide a thief. Players have their own colors: Brad (red), Tim (yellow), Clair (purple). At the bottom of the screen they are menu controls.

---

<sup>5</sup> <http://www.tourality.com/what-is-tourality/>

<sup>6</sup> <http://www.tourality.com/highscore.jsf>

<sup>7</sup> <http://www.tourality.com/how-to-play/challenges/>





Figure 3: Main game screen

In Tourality version 1.1, developers added a feature to generate game set according to user request parameters<sup>8</sup>. Parameters are the game mode (race, trail, chase, rush, act), amount of spots and the game type (single player, versus, team versus). They use an algorithm that randomly picks some spots from previously created online games by the game master.

## 1.4 GPS Mission

In this game, a player must complete so-called *missions*. Each mission consists of 2 to 12 *checkpoints* that must be passed sequentially. Distance to the next location on *radar*, suggestive photo and hints are displayed only after the previous one<sup>9</sup>. Radar is a code name of the mobile phone with running GPS mission software.

The mission usually consists of coordinates that the player has to reach<sup>10</sup>. However, in some cases, moving to the next checkpoint may require to answer a

<sup>8</sup> <http://www.tourality.com/game-sets.jsf>

<sup>9</sup> <http://gpsmission.com/>

<sup>10</sup> <http://gpsmission.com/gps-mission/Howto-Play.htm>

question such as *What is written in such a pointer?* and *What year this memorial is open?*[9].

Game master uses a web-tool to create a mission. He/she marks coordinates as checkpoints, attach photos as hints and add questions to the checkpoints. Game master also may hide bonus in the mission. This bonus is called *gold* (Figure 4A). By completing a mission the player earns a certain amount of points depending on the complexity of the mission) and bonus points. The players rank in the global leaderboard is decided according to his total amount of points.



Figure 4: Game Screens

The main feature of the game is that the player must find the answer to the question at the checkpoint. Automated game creation would require, a database of checkpoints would be needed, in the same way as in Tourality. In addition to this, questions and answers would be required.

In Table 1, we can see a summary of the games described before, their main features and the description of their main aim.

Table 1: Summary of location based games

	<b>Geocaching</b>	<b>Tourality</b>	<b>GPS mission</b>
Online/offline	offline	online	offline
Exact/relative	exact	exact	exact
Multiplayer		x	
Leaderboards		x	x
Search goal	x	x	x
Virtual/real goal	real	virtual	virtual
Education			x
Sport compete		x	
Generated levels		x	
The game mission	Find physical object with geocaching mark	be faster than opposite team and reach more goals	Visit all goals in the order and answer correctly for questions

## 2. O-Mopsi

O-Mopsi is a mobile orienteering game, where the player must reach all goals as fast as possible [10]. It has mobile software that allows a player to play game on mobile device.

### 2.1 Game rules

O-Mopsi game (Figure 5)<sup>11</sup> is defined by the list of the goals and title of the game. Goal considers information about location and representative pictures, also goal could have the title to give short description of the goal.

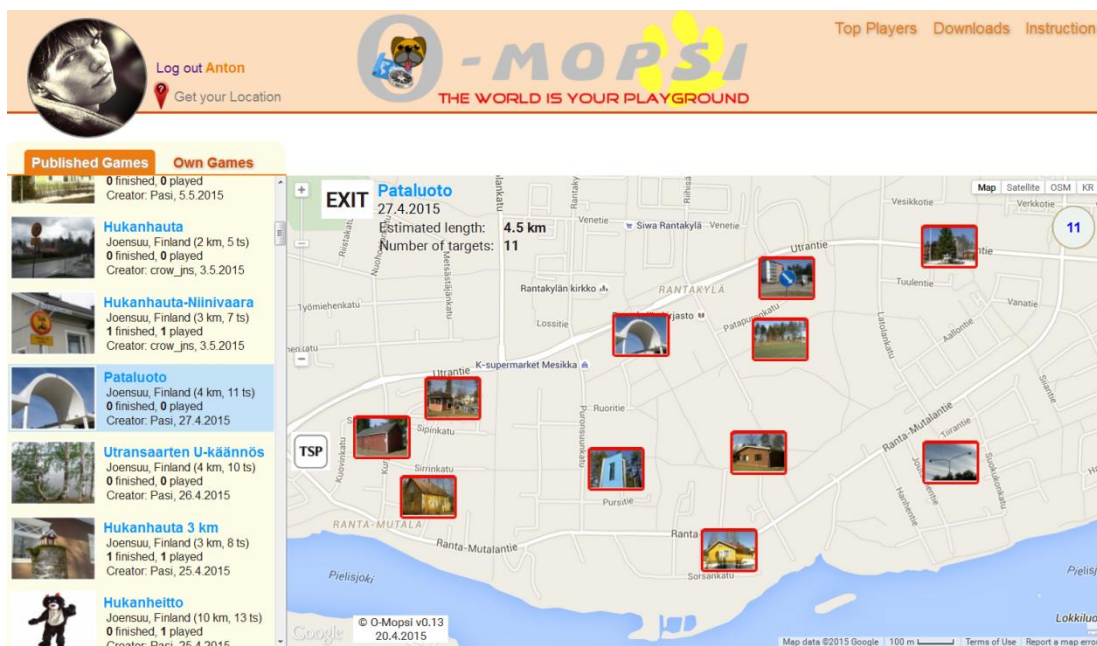


Figure 5: Game example

O-Mopsi games are for single player. In the beginning, player chooses the game from a list showing all available games sorted by their distance to the player's location. The goals become visible to the player only when he/she starts the game and clock starts ticking. Before that, player can only see the bounding region where the goals are. The Goals can be visited in any order. A goal becomes visited when distance relative to the player is less than 20 meters. This threshold was chosen taking into consideration GPS inaccuracies<sup>12</sup>. Players are ranked in the order of the completion time. Player can stop a game and continue it later, or he/she can cancel the game if too difficult. If continued later, the clock keeps on ticking and total time counted as a result. Game is over when all goals have been visited.

<sup>11</sup> <http://cs.uef.fi/o-mopsi/>

<sup>12</sup> <https://strava.zendesk.com/entries/21443922-Why-is-GPS-data-sometimes-inaccurate->

## 2.2 How to play

*Orienteering* is type of sport in which participants use a map and compass to reach control points, located in unfamiliar terrain (Figure 6). The participant who has the shortest time, or most of points is the winner<sup>13</sup>[11].



Figure 6: Player visiting control point.

Orienteering requires navigating from point to point in a predefined order using a map and a compass<sup>14</sup>. Unlike orienteering, the goals in O-Mopsi can be visited in any order, encouraging players to develop different strategies for selecting the order of visiting the points. The software shows the nearest goal by blinking it on the screen. However, this may not result in the shortest overall tour but gives merely a suggestion. Orienteering is focused on identifying the player's own and target location on map. O-Mopsi provides additional hint in the form of photo of the goal, and shows player's location on map obtained from GPS.

In Geocaching, the time does not matter and the goals are not grouped to form games or other entities. Orienteering and geocaching use physical objects, whereas O-Mopsi uses virtual goals (Figure 7).

---

<sup>13</sup> [http://epublications.uef.fi/pub/urn\\_nbn\\_fi\\_uef-20141098/urn\\_nbn\\_fi\\_uef-20141098.pdf](http://epublications.uef.fi/pub/urn_nbn_fi_uef-20141098/urn_nbn_fi_uef-20141098.pdf)

<sup>14</sup> <http://orienteering.org/about-orienteering/>

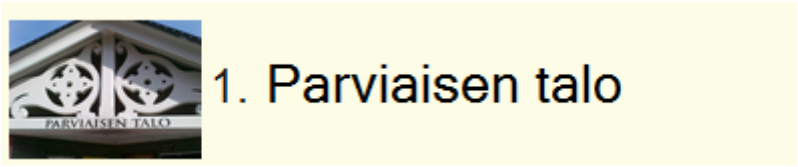


Figure 7: Example of a goal

O-Mopsi requires navigation skill. The total distance travelled, the choice of the starting goal, or the order of visiting the goals do not affect ranking. However, these might be used as criteria for alternative ranking. Finding the shortest tour would be very challenging, but GPS errors would add too much randomness to the results.

The *goal* is identified by its location, photo and a short description. The goal should be a real object in real world. It is possible to use also artificial goals but then the playing would simplify just to finding the location. The photo could also be just a hint, puzzle or a question for which the location gives the answer. To get the best result player has to be visit all goal as fast as possible. The ranking is based on the first trials, in other words, the second and later trails do not affect on the rank (Figure 8). However, if a player repeats playing the same game again, the new results will be counted on the *total* results list.

Player	Start	Time
Jukka	1.5.2014 07:58	23:44
Pasi	12.4.2015 11:10	24:05
Radu	1.5.2014 16:16	42:27

Figure 8: Game leaderboard

Although O-Mopsi is introduced as a game where players compete against each others, it can also be used as a sight-seeing tour for tourists when they arrive to a new city. Game helps to explore the city and show interesting places for the player[12]. For competitive people, O-Mopsi is tracker software that records the tours and maintains ranks. Other people might not be so motivated to go outside to exercise and they prefer to play video games inside. However this kind of people like challenges, and O-Mopsi is therefore a good opportunity for them to start exercising as well.

Game results can also be published on Facebook. It is likely that even local players may find interesting places in their home city; places where they have not yet visited. For people who are interested to play with their friends, the game provides good competing environment. Be better than others and sharing result via Facebook motivates friends to reach even better results.

## 2.3 O-Mopsi mobile software

O-Mopsi is available for Symbian, Windows Phone, Android and iOS devices<sup>15</sup>. In this thesis, we describe how to use O-Mopsi on an Android device. Device should have the screen at least 320x240 pixels and 16 megabyte memory.

After starting mobile software player will see buttons *games*, *login* and *settings* (Figure 9). If player has not login yet then the behavior of the games and login buttons will be the same, they will both forward player to the same login activity.

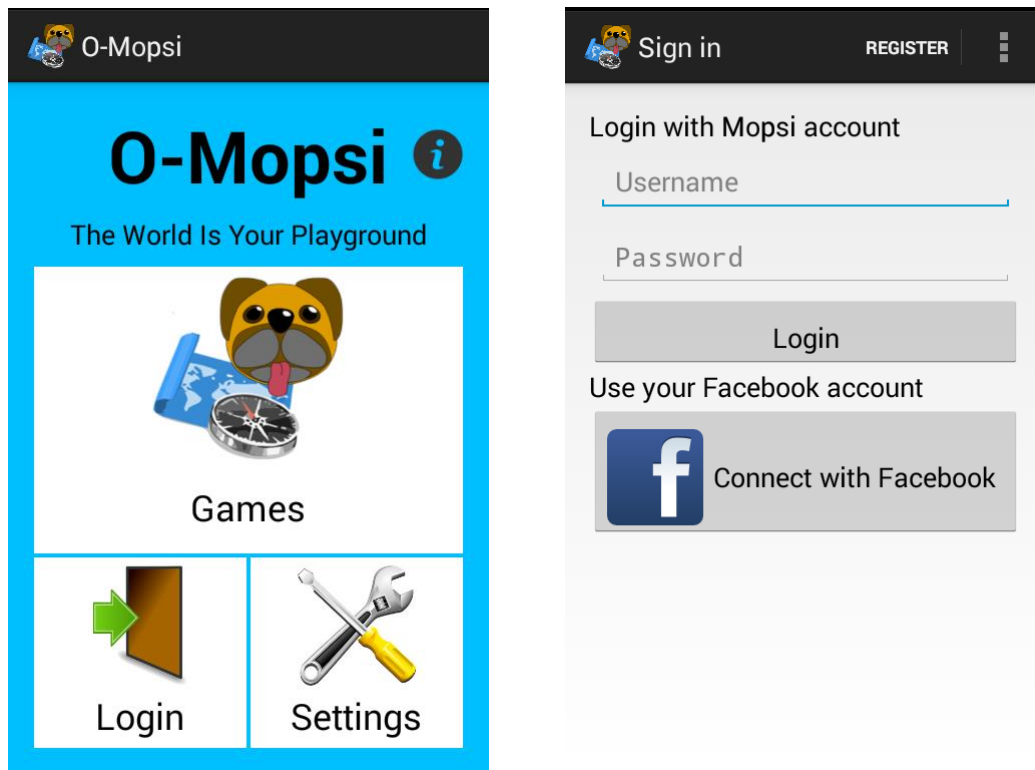


Figure 9: O-Mopsi software start screen (left) and login screen (right)

The player cannot start a game without logging into the software. Player may create an O-Mopsi account or use Facebook to log in. Players choose a game to play from the games list (Figure 10). The games are sorted according to their distance to the player. The distance is calculated from player's current location to the closest side of the bounding box of the game. In case the player is inside the bounding box the distance count is zero meters (Figure 11). Bounding box is computed by taking maximum and minimum values of the latitude and longitude from all goals of game. The games with the same distance are sorted in reverse order of creation date. The list also provides game *status*. Once a game has started its status becomes active. While game is in active status its playing time is cumulated until the game will be finished or canceled. Active games in the list have a *continue* button. When the player has chosen a game he/she will see a bounding box (Figure 10), and not the actual location of the goals. The reason for this is that strategy planning:

---

<sup>15</sup> <http://cs.uef.fi/mopsi/mobile.php>

deciding the visiting order of the goals is part of the gameplay and should consider the playing time as well. Player can also check the leaderboard and make decision if he/she wants to play or not. Location of the player is also shown.

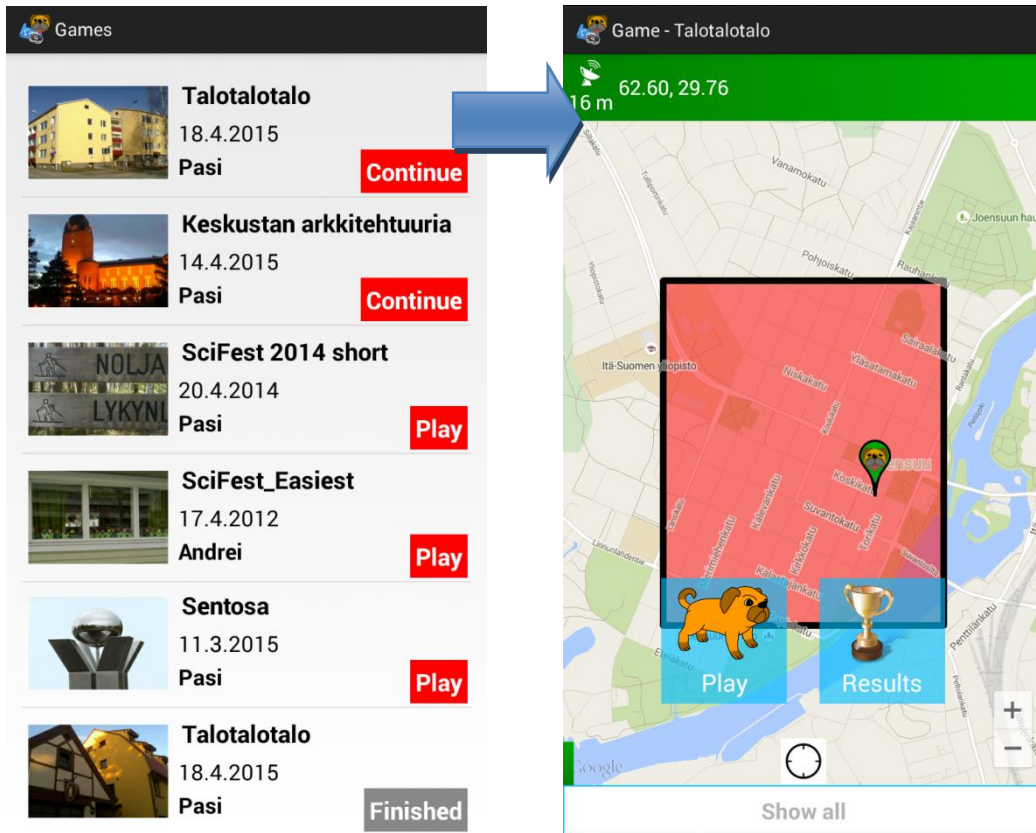


Figure 10: Game list (left) and Game activity (right)

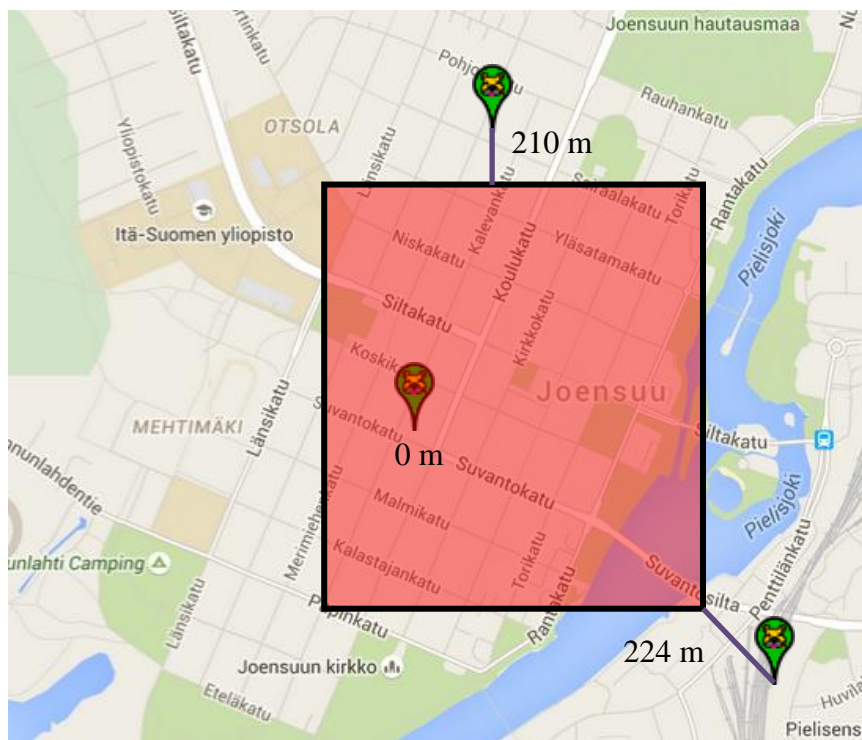


Figure 11: Distance from player is location to the bounding box

Software determines player location by using location services of the cell phone. O-Mopsi classifies the location based on its accuracy (Figure 12). *Yellow* color indicates that the software uses location from cell phone towers; *green* color indicates that the software uses location from mobile GPS receiver. *Orange* color indicates previous known location but that the current location is unknown. Player is recommended to wait for the green signal in order to play the game.

 Latolankatu, Rantakylä, 27 m Joensuu, Finland	Yellow
 Latolankatu, Rantakylä, 27 m Joensuu, Finland	Orange
 Latolankatu, Rantakylä, 30 m Joensuu, Finland	Green

Figure 12: Player location status

When player starts the game, exact location of the goals is shown on the map. Player can then to decide in which order he wants to reach the goals. Player may pan and zoom the map (Figure 13), and switch to a mode which zooms the map so that is shows player location and the closest unvisited goal but nothing more. In this mode, the map will be automatically zoomed in/out to keep both player and the nearest goal on the map screen.

On the game screen, the following statistics are shown:

- Time spent in the game since last login,
- Distance he passed in the game since last login,
- Number of goals the player have already visited,
- Distance to the closest goal from the player.



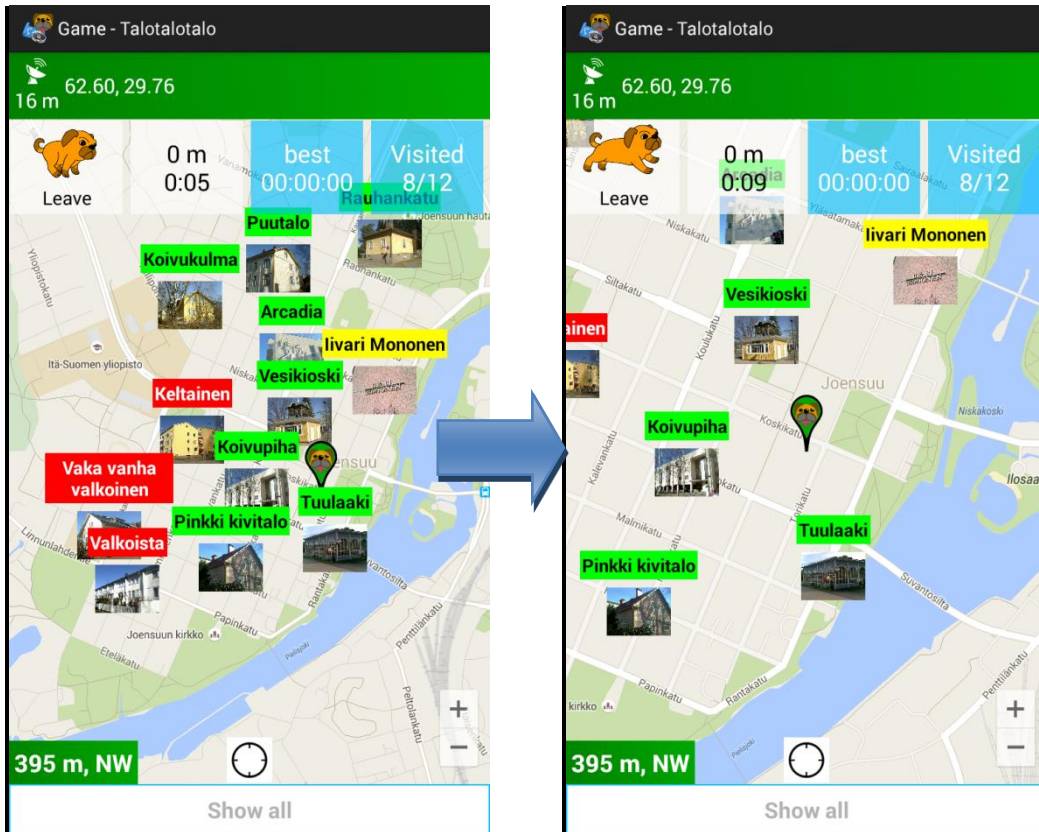


Figure 13: Changing map view

On the map, goals have different colors: *green*, *yellow* and *red* (Figure 14). Green means that the goal has been visited. Yellow is the closest unvisited goal, and red is an unvisited goal. Player can also see the direction and distance to the closest goal. The closer the player is to the goal the more frequently the software plays beep sound. When distance between the player and the goal is less than 20 meters, the status of the goal changes from *unvisited* to *visited* and the software plays *Ta Da* sound. When visited last goal it plays applause sound. Player may also check the goals on a list (Figure 14), which contains two groups of goals: unvisited and visited. Both groups are sorted by the distance between the goal and player's current location.

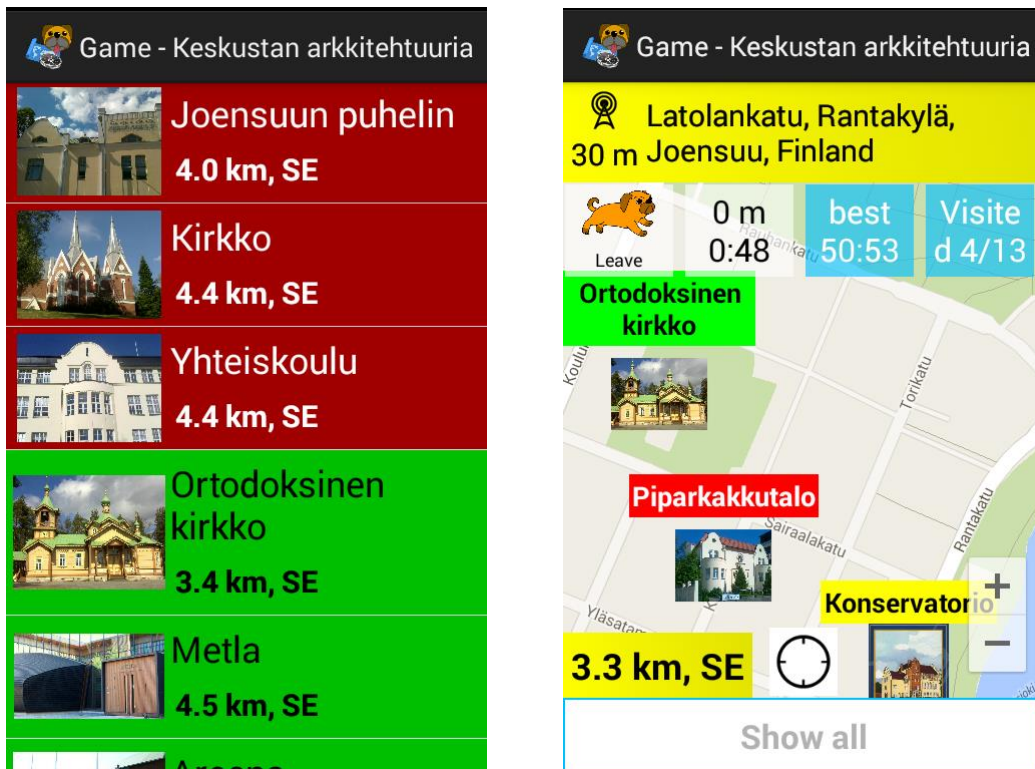


Figure 14: Goals list (left) and goals on the map (right)

When player reaches the last goal the game is over and the result is shown (Figure 15). The results screen shows the list of all players who have finished the game ranked according to the competition times.

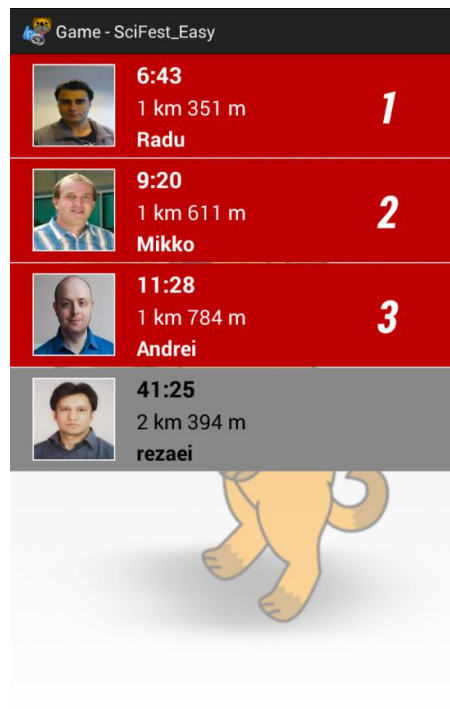


Figure 15: Results screen

### 3. Creating a game

In O-Mopsi a new game can be created by selecting a set of goals and giving a name to the game (Figure 16). A goal can be added from various sources:

- Mopsi user photo collection
- Mopsi service collection
- Uploading own photo

When game master uploads own photo he/she needs to setup position of the goal on the map. Once all goals have been added, it is possible to edit them when needed. Game master can update a description for the goal; simply click on the name (Figure 16). Most important feature is to change the position. This is done by clicking to the position marker (Figure 16) in the goal list and dragging the marker on the map to its new location. Adding goals from Mopsi collection is the simplest way, because location is already defined and it may have a title already[13]. A goal can be removed by clicking on the remove button (Figure 16).

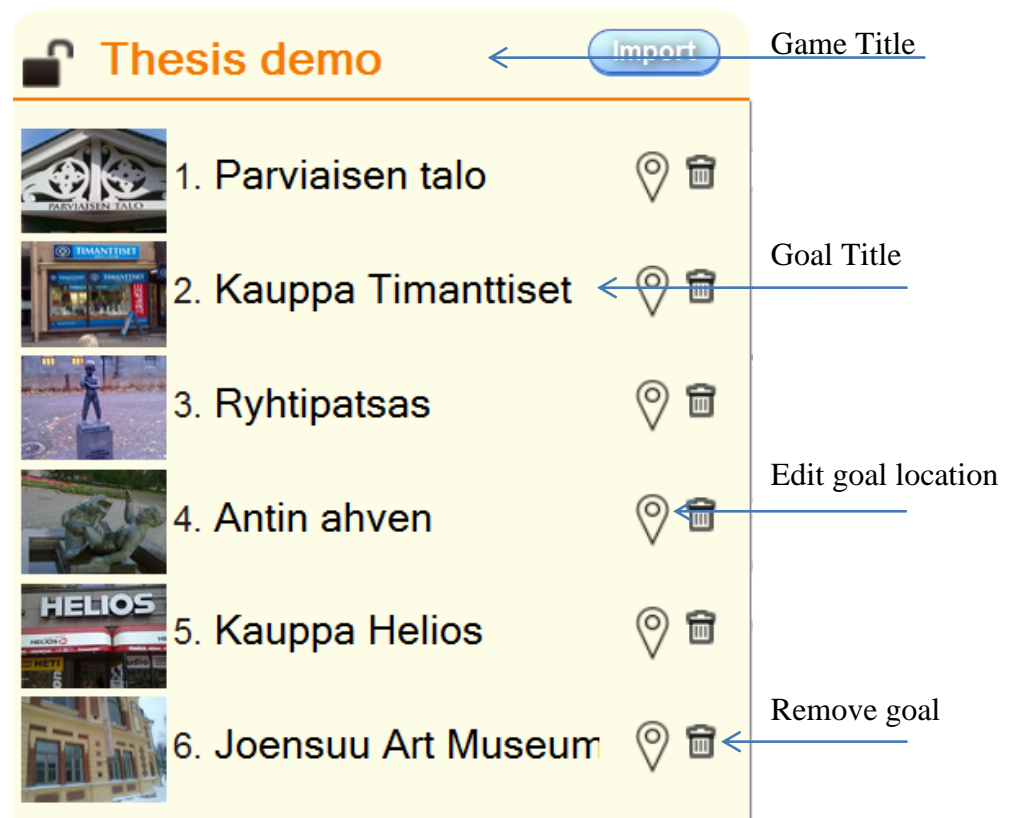


Figure 16: Edit game controls

When the game is ready the game master can check an approximate for the *reference tour*, which connects all the goals (Figure 17). According to the distance of this tour he can decide to add or remove goals depending on the difficulty level he/she has in mind for the game. When the game is finally ready the game master will publish it. A published game can later be unpublished by the game master but after someone has completed the game, the game goals cannot be edited anymore; either location either title. Any changes in the game can make it easier or harder, it is unfair to the players that have already been played this game.

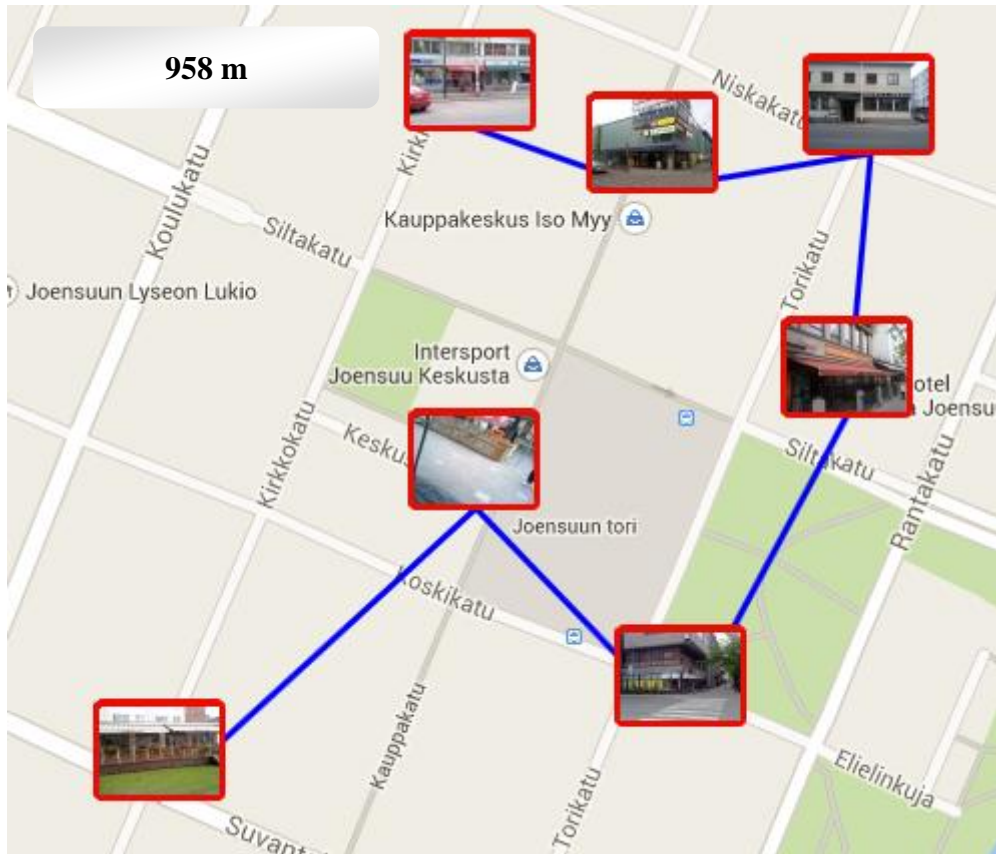


Figure 17: The reference tour of the game.

Any user can create new games by the following 6 simple steps<sup>16</sup>.


1. Log in to O-Mopsi
2. Click *Published Games* tab
3. Click *New Game* button
4. Enter the name
5. Add goals
6. Click *Exit*

O-Mopsi start page allows game master to see the list of already published games, own games, top players, instruction how to use O-Mopsi and download software to his/her smartphone (Figure 18). When player opens *Own Games* tap, he/she can access all games he/she has ever created. New games can be created, and old ones removed and edited (Figure 19).

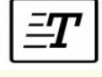
---

<sup>16</sup> <http://cs.uef.fi/o-mopsi/helper/index.php>


**Published Games**   **Own Games**




**Siltojen taidehökkiä**  
Joensuu, Finland (2 km, 8 ts)  
0 finished, 0 played  
Creator: Pasi, 16.6.2015




**Texty**  
Joensuu, Finland (0 km, 6 ts)  
0 finished, 0 played  
Creator: Radu, 15.6.2015




**Andorra in Nutshell**  
Andorra la Vella, Andorra (2 km, 1 ts)  
0 finished, 0 played  
Creator: Olli, 28.5.2015



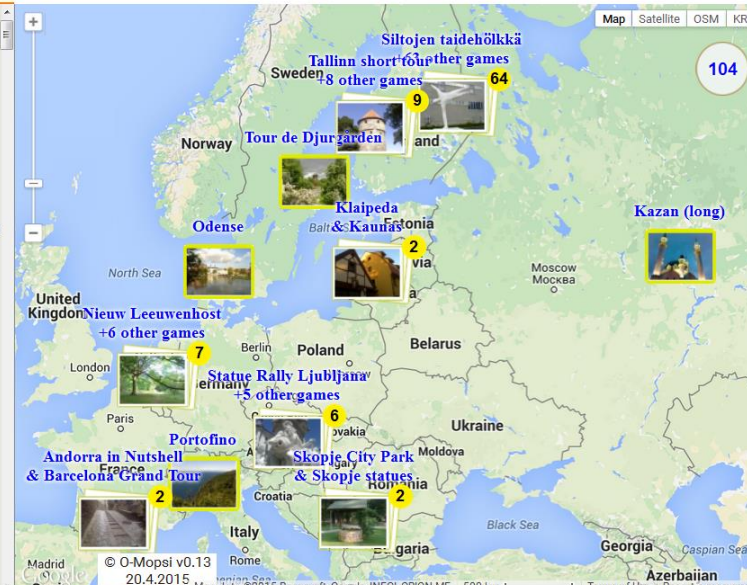
**Reijola kuntokierros**  
Joensuu, Finland (5 km, 17 ts)  
0 finished, 0 played  
Creator: Pasi, 15.5.2015



**Reijola keskimatka**  
Joensuu, Finland (2 km, 3 ts)  
0 finished, 0 played  
Creator: Pasi, 15.5.2015




**Reijola Toukoviisi**  
(0 km, 5 ts)  
0 finished, 0 played  
Creator: Pasi, 15.5.2015




Map controls: Map, Satellite, OSM, KR. Scale: 500 km. © O-Mopsi v0.13 20.4.2015. Map data ©2015 Basarsoft, Google, INEGI, ORION-ME.

Figure 18: O-Mopsi start page


**Published Games**   **Own Games**




**testCombination**  
Joensuu, Finland (2 km, 6 ts)  
0 finished, 0 played  
16.6.2015




**Test distance (Editing)**  
Joensuu, Finland (2 km, 6 ts)  
0 finished, 0 played  
5.6.2015




**Thesis demo (Editing)**  
Joensuu, Finland (< 1 km, 6 ts)  
0 finished, 0 played  
5.5.2015



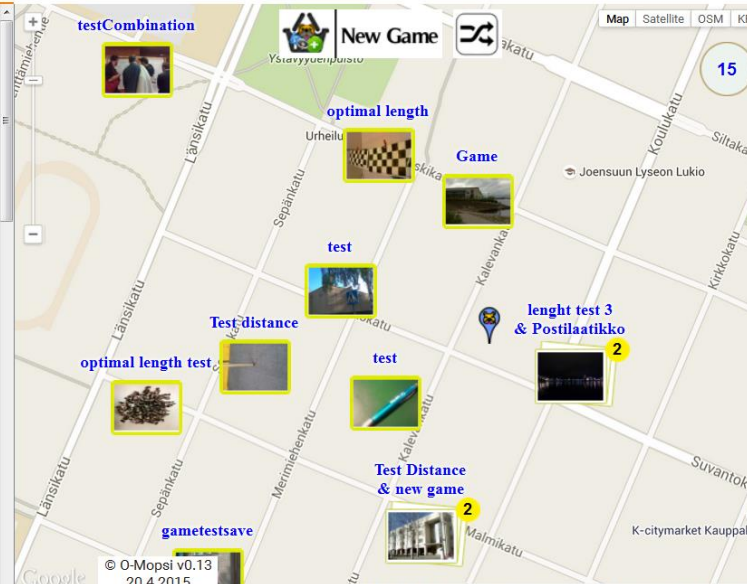
**gametestsave (Editing)**  
Joensuu, Finland (< 1 km, 1 ts)  
0 finished, 0 played  
5.5.2015



**mopsitargettest (Editing)**  
Joensuu, Finland (< 1 km, 6 ts)  
0 finished, 0 played  
27.1.2015



**test (Editing)**  
Joensuu, Finland (< 1 km, 2 ts)  
0 finished, 0 played  
27.1.2015



Map controls: Map, Satellite, OSM, KR. Scale: 50 m. © O-Mopsi v0.13 20.4.2015. Map data ©2015 Google.

Figure 19: Own games screen

When game master clicks *New Game* button it initiates the game creation process. After this he/she needs to enter the name of new game as shown in Figure 20. The name should have at least 3 letters.

Figure 20: Game name.

When new game is created the user can add goals to the game. Game master can use own Mopsi photos, friends Mopsi photos, Mopsi services (Figure 21) or upload photo from other sources (Figure 22)[14]. In case of using other photos, game master must specify the location of the uploaded photos on the map.



Figure 21: Controls for game creation

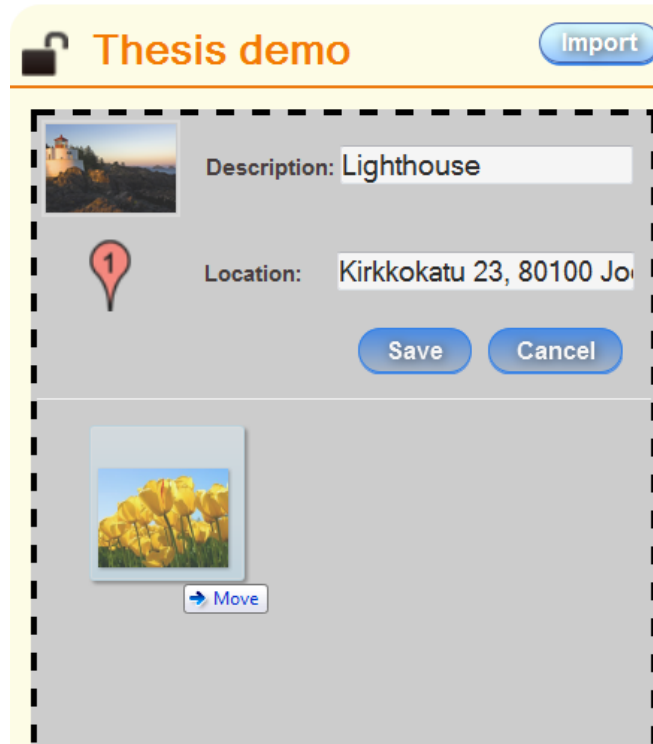


Figure 22: Add photos from computer

After selecting a photo and in popup window one needs to click *Add Goal* button (Figure 23). It is also possible to add a goal simply by right-clicking on it. Game master can exit from the game and continue editing the game later. When game master has added enough goals, he/she can then share this game by pressing the *publish game* button (Figure 24).



Figure 23: Information of the goal



Figure 24: Goals list



## 4. Criteria for a good game

Quality of the game depends on several factors: primarily on the quality of the goals, but also on their location relative to each other. The following factors affect the quality:

- Photo quality
- Location
- Name
- Size
- Accessibility

Photo should be sharp and have high enough resolution. Object from the photo should be clearly recognizable, located on the center of the photo and there should not be other objects covering the goal[15].

A goal should be located in open air area and its location on map should match its real location (few exceptions will be noted later). Name of the goal should be short but yet clearly describe the object or represent the location. Short names do not overwhelm the map and are easier to read. Size of the goal can vary, from small to big buildings, but the goal should be visible from 20 meters away. A goal should be accessible at any time of the day and year. The player should not have any problems to access it during playing.

Examples of goals for which the location needs to be tuned are shown in Figure 25. There are objects that are far away from the places that the pictures have been taken. It is not understandable where the location of the goal is; where the object is or where the photo has been taken. This is illustrated in Figure 25. It is unclear to which colorful balcony player has to go (Figure 25). In the other example player needs to be on the island to reach the goal but the goal captured in the picture is on other side of the road (Figure 25).



Figure 25: Distant objects

When user adds a photo to O-Mopsi collection, the location should be checked and tuned when needed (Figure 26). Single Spruce is easy to identify far away, as players can see the object well before it will be reached. If the object is easily reachable the location should be on the center of the object. For example bridge over the river. In case if it have nice view to the object and it has only one way to reach the object location could be stay on the view point to the object such as the single building in the forest. A cluster of objects is also fine if not too large. All the objects describing the goal should be within 20 meters in range. In case of large clusters player cannot guess from which side he/she has to approach to the object.

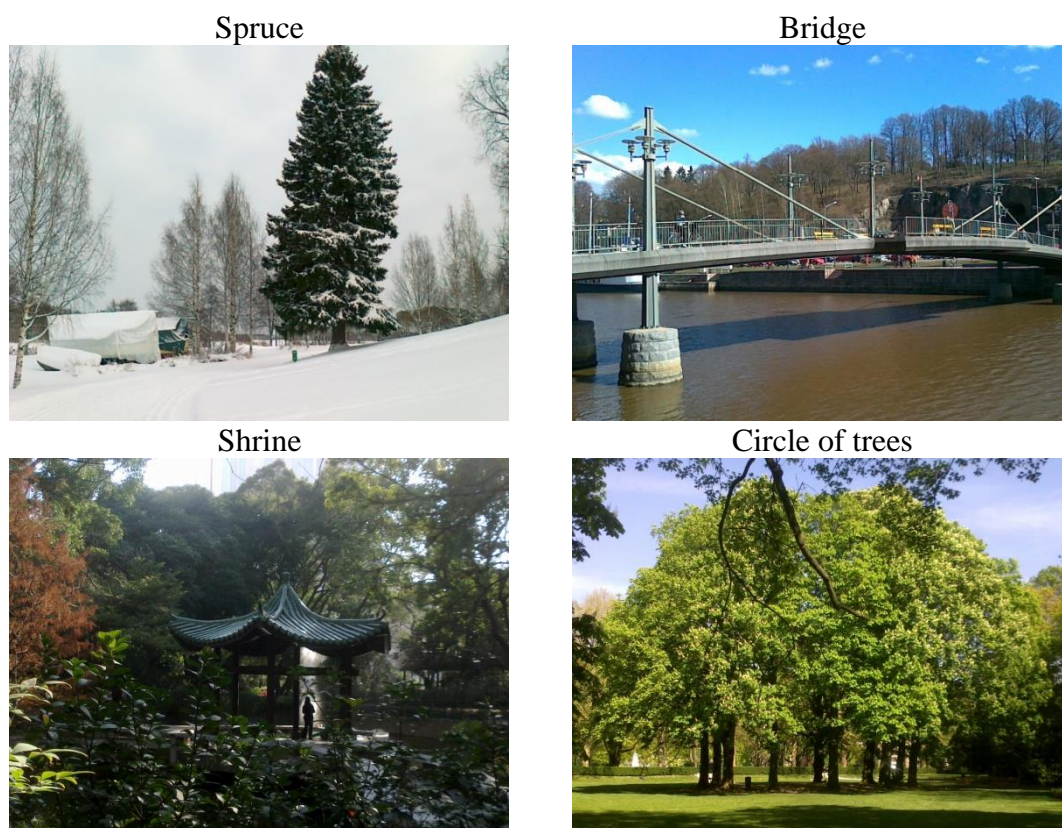


Figure 26: Examples of goals

A goal should have a name. The name helps player to understand what exactly he/she is looking for (Figure 27). The name should be short and clear. When using Mopsi targets (services or photos), the already existing description might need to be edited. For example *Metla campus's parking place for bicycles* should be renamed to *Bike park* or something else that is short but informative. Another example when one needs to change the name of the goal is the *Jogging route landscape* which could be changed to *Traffic sign* because it is the most visible target on the picture even though the author of the photograph has described the context when the photo was captured, and not what is in the picture.

Metla campus's parking place for bicycles



Jogging route landscape



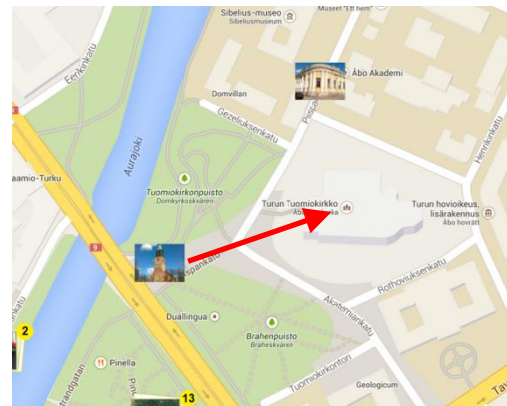
Figure 27: Rename goals

Buildings could be reached from multiple sides, and it is not clear where exactly the goal located. Game master should therefore follow few simple rules. Find recognizable object on the building such as art, picture or sculpture as on the building (Figure 28). Use the main entrance as location of the goal if nothing else interesting is around. Always tune the location to the object in the photo (Figure 28). Picture may be taken far away from the building and player may spend extra time around the goal searching for the proper location of the goal. Pictures may be taken from the closest spot near to building, by looking up to indicate to the player that the goal is a building and he/she should come close to the building.

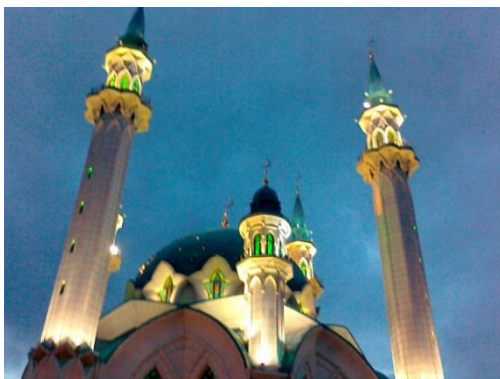
Cathedral



Tuned location



Col. Sherif



Hotel



Figure 28: Buildings as goals

In case the building is too big (Figure 29) and it is not clear where the player should go to then it is better choose an object near the building, or take a photo only of a part of the building.

A. The picture in Figure 29B could be a goal, but it is better to use an object is located near the main entrance,

C. Church in Figure 29D is recognizable object, but it is rather large so if you do not use the main entrance as a goal then better to use a picture of a small object or small part of the church.

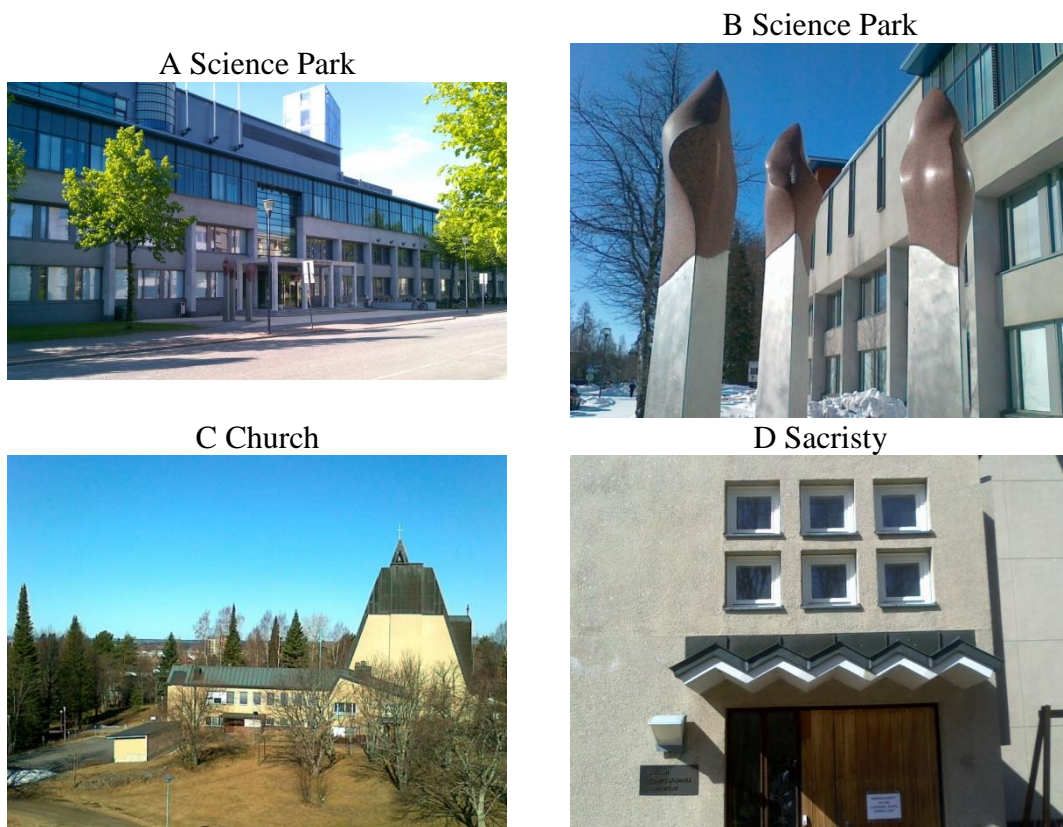


Figure 29: Big buildings

Objects like signs (Figure 30 A, C), statues (Figure 30 B, D) and arts (Figure 30G) provide information to tourist and help to study the city. They are good goals. Also in the town a lot of small recognizable objects also might be of interest to players (Figure 30 E, F, and H).



Figure 30: Good goals

Too small object can be a bad goal, because in most cases the software will notify the goal is reached before player will see it (Figure 31A, B). Another possible situation is that the goal may be moved from previous position or grow-up to look different (Figure 31C).

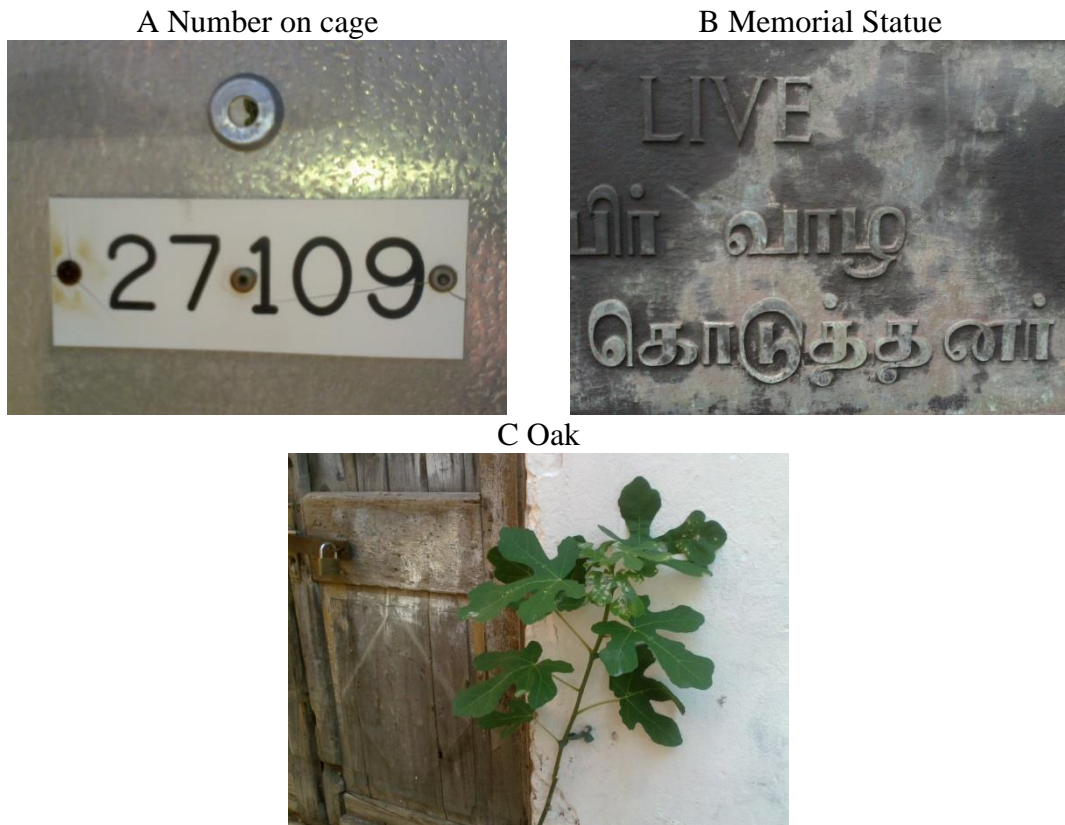


Figure 31: Too small goals

Points of interest could be a park or entire lake (Figure 32). Then the game master should choose a unique view that identifies a detail, or place that cannot be missed (trail leads directly to this location).

- A. Single trail on the top of the mountain with nice view on the cape,

B. In case water spring it is clearly understandable where the goal is located with the nice view,

C. The object could be not visible on the picture but the name should provide additional information what the player has to search. Thus, the picture and the goal name together describes the place to find,

D. It looks graffiti but it is park logo. The point is that the park is large but the entrance has clearly visible sign including the name of the park.

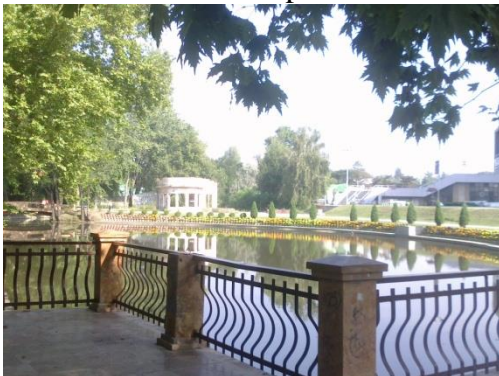
A Portofino



B Water spring



C Under Maple tree



D Graffiti



Figure 32: Views and Areas

Before using a goal in the game the game master has to think whether it is reachable any time (Figure 33). The photo could be nice and have correct location, but could be reachable only part time. Those kinds of goals should be moved from unreachable location to the main entrance, for example.

A. Good goal that represents the pond inhabited by flamingos. The game master wants to use this as the goal then better to locate it on the viewing point because player cannot access the flamingos or could be illustration as a zoo goal.

B. Hut is good goal in summer time, but in winter it can be more difficult to reach,

C. Nice goal but make sure the goal is located on the main entrance, and not in the middle of track, as it might not be reachable (at least on race days access is strictly forbidden).

A Flamingos



B Hut



C Horse race track



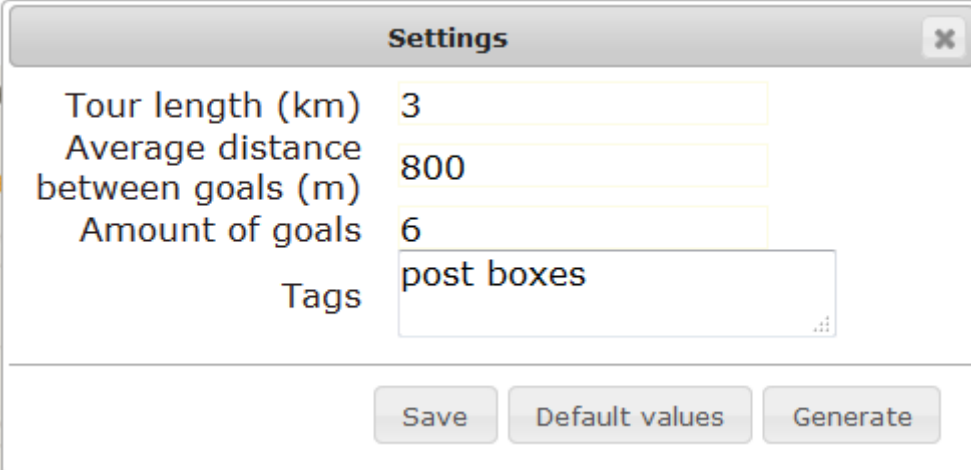
Figure 33: Accessibility

In short, a good goal is such that the player can clearly recognize the object and it accessible. The object can be a statue, museum, theater as they are interesting places to see by visitors from another city. Touristic places that have the spirit of the area are especially good (scientific art in university campus, wooden sculpture near the museum of woods). The goal should be possible to visit at any time of the day. If the goal is a large building, it should have goal positioned on the main entrance or one of the arts on the wall or recognizable object near the building. Goal could be a nice view from place where the player could arrive by single trail.

## 5 Automatic game generation

O-Mopsi can be played anywhere in the world. But to be able to play a game, it must be located near the player. In case there are no games created near the player, he/she will quit the game and might not use the software anymore. It is therefore important to have game available for the player when he/she starts the software. To create a game, the game master has to spend time to collect materials to have good goals. Game master would benefit automatic game creation to save time to create game. He/she can edit it if not satisfied with the final result. It is the hardest part of the development to provide the satisfied result of the game. O-Mopsi has got only limited amount of games. If player already finished all of the ones nearby, and he/she wants to play more, automatic game generation would provide a game for such an active player. For these reasons we develop automatic game creation feature in the software.

Game master can use generate game button instead of create new game and get ready game. In the game settings menu (Figure 34) it is possible to change game parameters: tour length, average distance between goals and amount of goals and goal tags. It is also possible to change the current location to add goals in another area.



Parameter	Value
Tour length (km)	3
Average distance between goals (m)	800
Amount of goals	6
Tags	post boxes

Figure 34: Generate game settings

Amount of goals is the maximum amount of goals which will be in the game. If not enough data in the area the number could be less. Average distance between the goals and the tour length are also more like guidelines. But the algorithm will return the closest result that it finds. Tag is a single word which describes the goal; game master can use space or comma to separate tags. If request contains multiple tags then the goals that will be used must have at least one tag matching to the title name.

After pressing the generate button, the automatically selected goals appear on map (Figure 35). It is possible to regenerate a new game if game master is not satisfied with the game provided. When game master is satisfied with the game he/she *saves* the game. When the game is saved the goals can then be manually modified or removed. New goals can also be added manually.



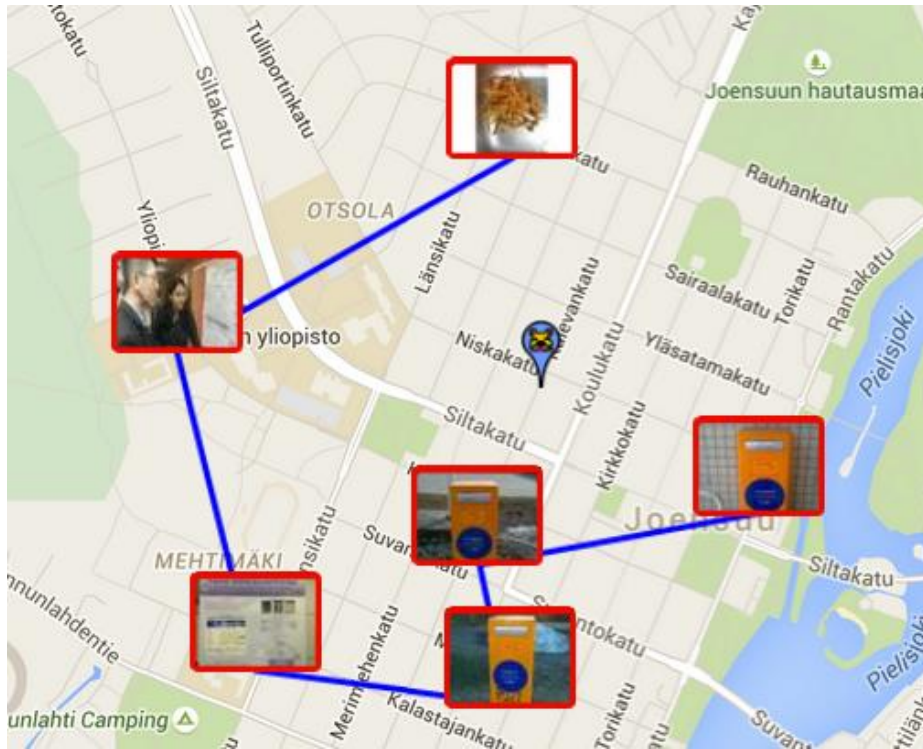


Figure 35: Example of generated game.

The algorithm has the following input parameters: coordinates of the game location, tags (optional), number of goals and the average distance between the goals. Those that are not specified will have default values. General principle of the algorithm is to find goals around the location and then use a heuristic to reduce the number of the candidate goals, and finally perform a brute force algorithm to select the best combination of the goals.

When the player asks for a generated game, the software uses default parameters. These are: five goals, 150 meters average distance between the goals, two kilometers length of the tour, and no tags used in the selection. The software then takes the current location of the game master and runs the Algorithm (Figure 36) with the following steps:

- 1) Get all candidate goals around the location, that match the given tags
- 2) Sort the retrieved goals according to their quality
- 3) Generate a combination of the goals that fits the parameters.

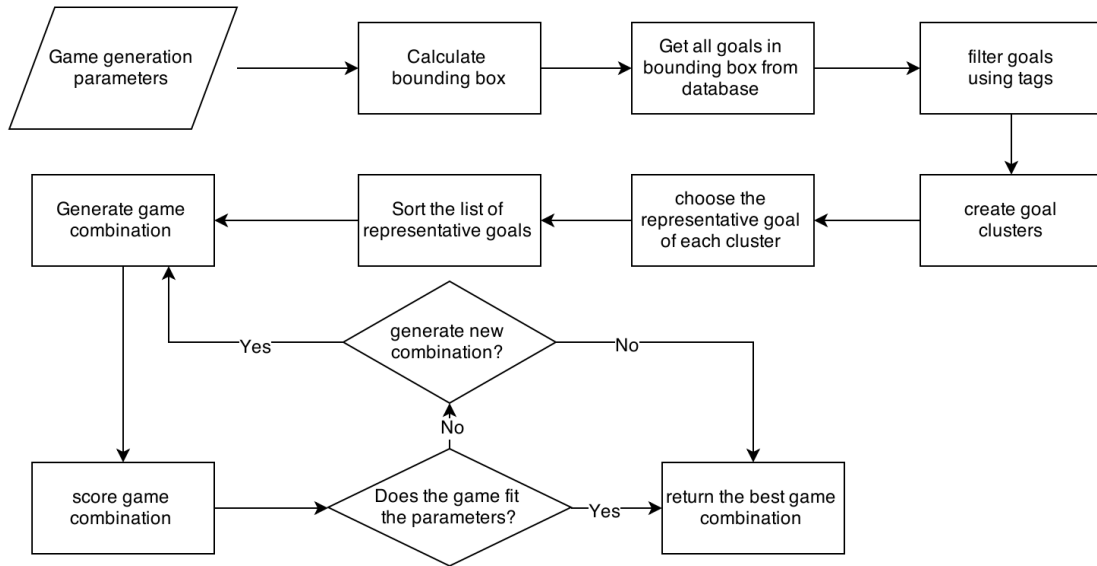


Figure 36: Algorithm flow

The brute force algorithm first creates a combination of the goals, and checks whether it fits into the game parameters. If it fits then we have found a good game to play. If it does not fit, then the game is scored and saved (in case it will be needed later anyway), and next combination is generated from the candidate goals. If no fit game is found after 10,000 trial games have been generated, we then select the game with highest score among the ones that were stored. The value 10,000 was chosen just to prevent the brute force algorithm looping forever.

## 5.1 Finding possible goals

Database contains all possible goals for a game. Some of them are too far to the player from the location according to the expected tour length. For this reason we first filter the goals. According to the estimated tour length we build a bounding box around the location. The length of each side of the bounding box is  $\sqrt{2} \cdot x$ , where  $x$  is the wanted length of the tour. It is needed because player can request game from area without any goals in one part of the area in the bounding box. For example, water areas have usually no goals, so if the location is near the coast, it would still be possible to generate a suitable game. This is demonstrated in Figure 37, where the lines present possible tours, when two kilometers length has been requested. We restrict all goals inside this box. The algorithm makes SQL request to the database and receives an array of goals within the bounding box.

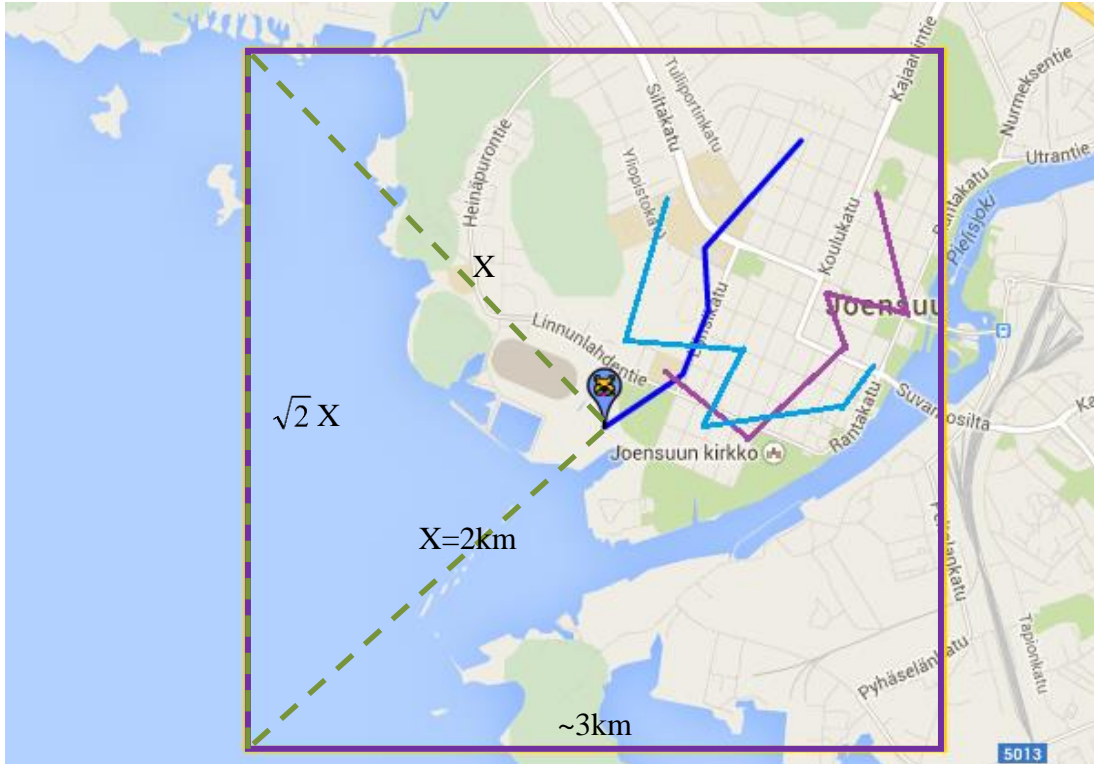


Figure 37: Bounding box for a game with expected length of 2 km

The last step of the filtering removes goals that have no relationship to the tags (if specified). Only goals whose title includes one of the tag words are accepted at this stage.

## 5.2 Automatic goal ranking

We use photos from Mopsi services and users' photo collections. Photos from the user collections do not always have good quality for the game, and some of them are located on the same position so that we should choose only one of them. For this reason, we measure how well a photo fits in this game. Photo has attributes:

- title
- geolocation
- location accuracy

According to this data the photos will be measured (Figure 38). We will check conditions:

1. Compare location accuracy.
2. Does the goal has title
3. Is the goal less than 20 meters away (from a road)



Figure 38: Display goal information

A good game contains good goals. Some areas contain large amount of potential goals. We therefore sort the goals before the brute force algorithm starts to combine a game from the goals.

The ranking of the goals uses criteria: *accuracy*, *location* and *name*. Those criteria were chosen because it is important for the goal to have accurate location. Accuracy criterion contains three states: *fixed*, *gps*, *cell*, according to the source of the location provider. Fixed means the user has tuned the location before or after uploading it to O-Mopsi. Gps means a location from GPS sensor of the mobile phone. Cell is location provided by cell towers, which devices use when GPS is not available. Location criteria is needed because player tries to reach goals as fast as possible, so he/she will not be happy if the goal not be accessible from the road (that goals are located more than 20 meters away from the road) or even not accessible at all. In this case player could not finish the game. The location criterion is *how far it is from the closest road*. Name criterion means *whether goal has title*. The goal that has a title is always better than a goal without it because it provides information to the player to give a hint what to look for. The game generation algorithm uses those criteria to rank the goals. In Figure 38, blue circles provide information about location source (accuracy criteria), red lines represent distance to the road (location criteria); black is the name of the goal (name criteria).

The most valuable criterion is the accuracy criterion. It divides the list of the goals into three groups: fixed, gps and cell, in this priority order. Next the criterion is the name; it divides each group into two subgroups: title or not. The last criterion is the location, which sorts each subgroup by the distance to the road, from the closest to the farthest. Those criteria help us to the rank goals and choose better goals first (Figure 39).



Figure 39: Goal ranking

The highest ranked goal will be with *fixed* location, having a title and located on the road. The lowest ranked goal will be with *cell* location, without title and more than 20 meters away from the road.

### 5.3 Goal clustering

There might be a situation when several goals are close to each other. In this situation player could reach these goals at the same time. To avoid this situation we cluster the goals and pick only one goal from each cluster (Figure 40).

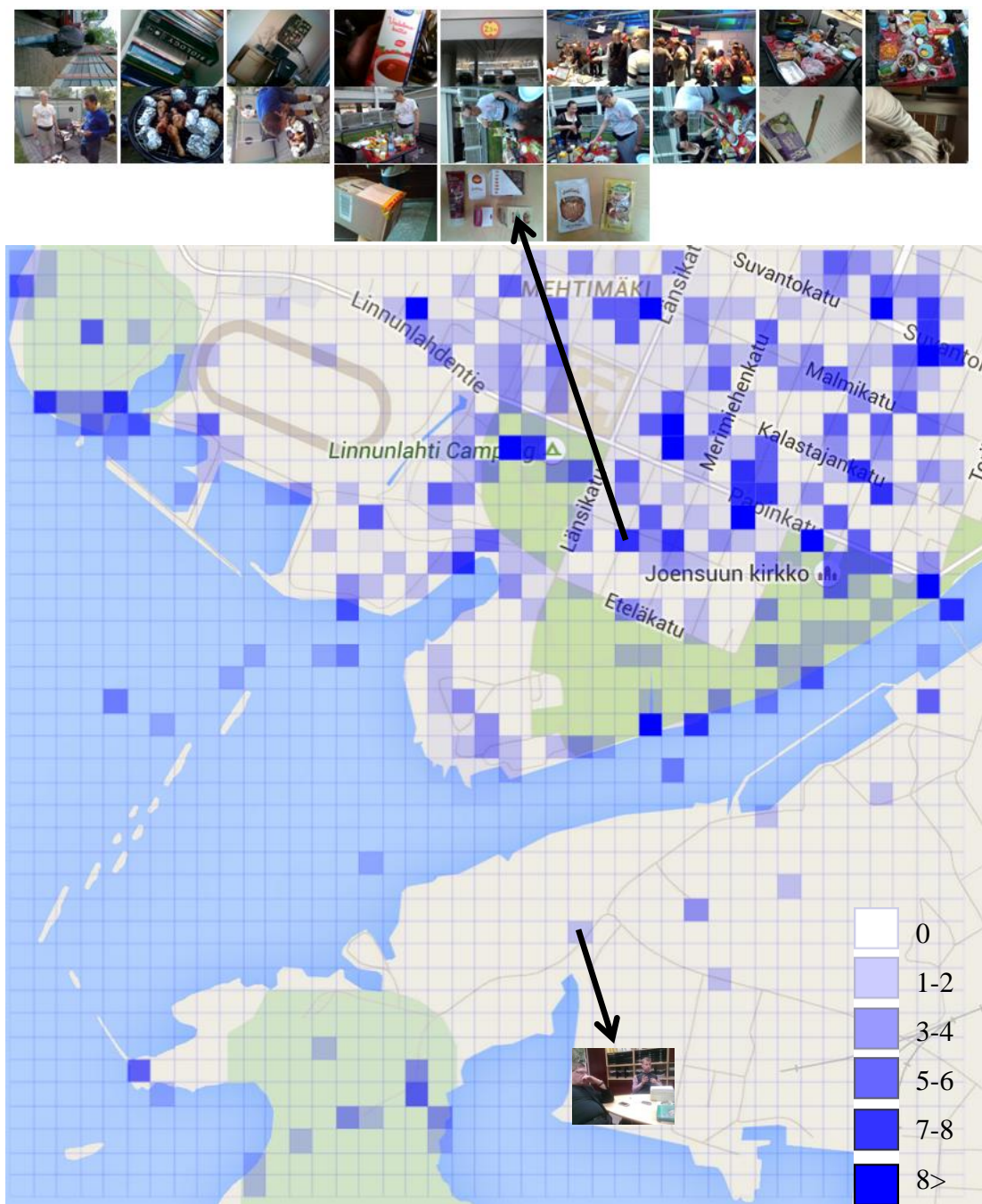


Figure 40: Clustering of the goals by 2d-grid. The number of goals in each cluster is shown by the color.

The player location is shown in the center. Cluster is a square with size of  $50m \times 50m$ . The algorithm fills the cells within the bounding box by assigning each goal to one cluster. All images in each cluster are scored according to the criteria, and the algorithm will pick the goal with the highest rank as the goal to represent the cluster.

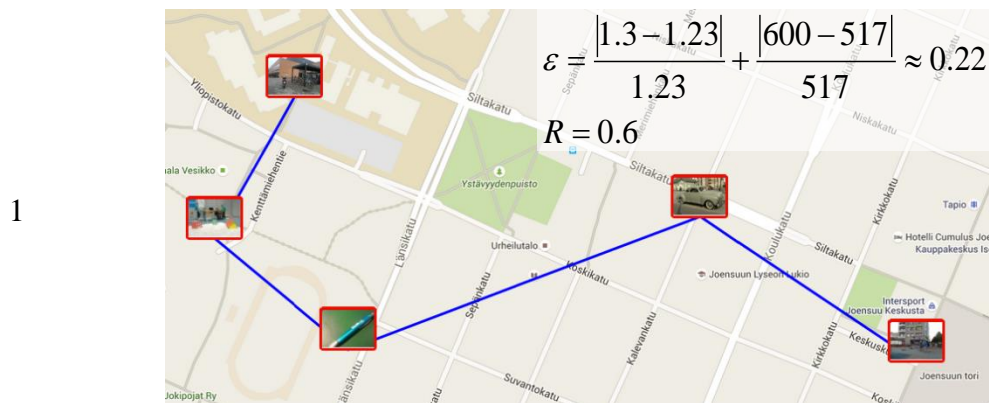
## 5.4 Evaluation of the quality of the game

The algorithm uses brute force to find more suitable tour to the given parameters. We generate a list of the tour candidates but we provide them only one at a time. Tours must be evaluated and they must be as close as possible to the given parameters. The algorithm uses tour distance and average distance as the parameters. The number of goals and tags cannot be used for tour evaluation. Goals that do not fit to the tags have already been removed from the candidate list of the goals. Algorithm provides exact amount of the goals if enough goals were found. If not enough then if a single tour is available and no need to evaluate it.

*Relative error* is the ratio of the *absolute error* of the measurement to the accepted measurement. Absolute error is simply the amount of physical error in a measurement<sup>17</sup>. Evaluation of the game is the sum of the relative errors (1) for the expected tour length and the expected average distance between goals. A smaller relative error means better the quality of the game. All tours that have relative error less than 0.3 are grouped in one list and inside this group we pick up the hardest game according to game complexity criteria. This threshold is chosen because it is not too noticeable that the distance of the reference tour is different from the expected distance of the reference tour. A hard game is more interesting for advance players, beginners will be motivated to try again to reach better results.

$$\varepsilon = \left| \frac{X_{measured} - X_{parameter}}{X_{measured}} \right| \quad (1)$$

For example, consider parameters with five goals, 1.3 kilometers tour length and 600 meters average distance between the goals. Algorithm provides several game candidates (Figure 41). The games number one, two and four have relative error less than 0.3. Algorithm chooses the game with higher R value from one of the games. The game number two will be provided to the player.



<sup>17</sup> <http://www.regentsprep.org/regents/math/algebra/AM3/LError.htm>

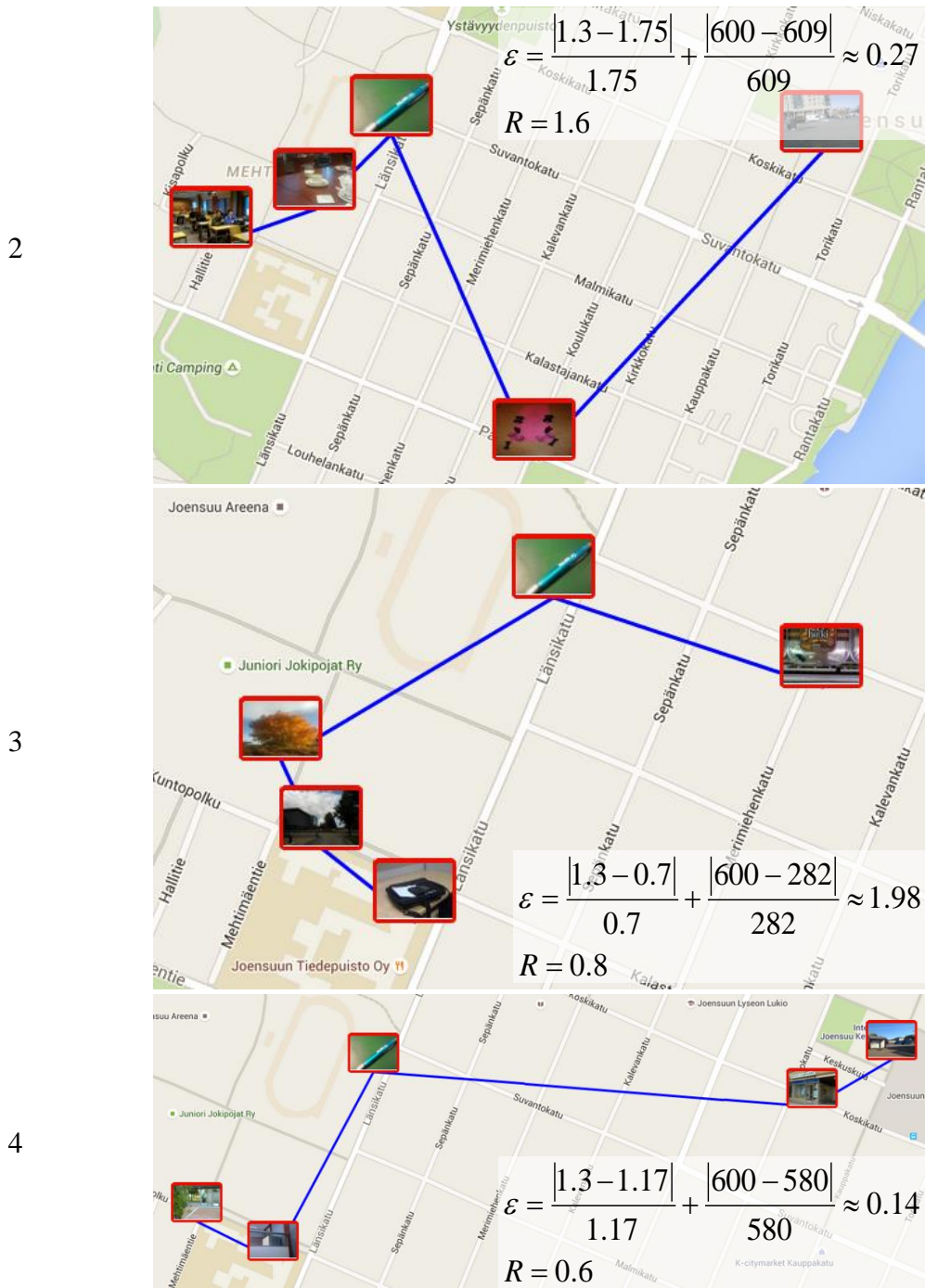


Figure 41 Game condidates

## 5.5 Game complexity

To finish a game with a good result player needs to visit all goals as fast as possible. To do this he/she needs to choose the tour of minimal length and follow it as fast as possible. Some games can be finished relatively easily, others are more difficult. For the players it is important to play non-easy games to keep interest to the game. To determinate how easy or how hard a game is, we define next the concept of *game complexity*.



When player starts a game, he/she must decide in which order to visit the goals. The player cannot see the location of the goals before he/she has started the game, so he/she must start at any place and hope the choice of starting place is a good one. Assume that the *lucky* place to start is a place to which the nearest goal is the start or end point of the shortest tour.

Using the location of the goals we evaluate the game complexity as follows. We can do it in different ways, including the lucky start, and how similar the reference tour is to a strait line. We used the following factors for the measurement.

- Player starts from lucky place
- Player starts from random place
- How close reference tour to the line
- Variation of the lengths of the tours
- Average mismatch between reference tour and greedy tour

Based on these factors, we consider next the following alternatives for measuring game complexity:

- MM – Mismatch criteria
- SSG – Same start with greedy strategy
- Av – Average criteria
- A – Angle criteria
- $\sigma$  – Standard deviation criteria

Assume that the player starts the game from the lucky place and will use *greedy* strategy to reach all goals. Greedy strategy is that the player always goes next to the closest unvisited goal. The game complexity is then defined as a ratio between the reference tour length and the length of this greedy strategy.

The reference tour is calculated using *Tabu search* algorithm. This algorithm been chosen because it provide results close to optimal and it is fast enough to be used on real time web systems[11]. We use the same start goal as initial goal for *greedy search* algorithm to find the tour for the greedy strategy. Distance between goals is measured as a straight line.

We use the equation (2) to calculate the game complexity. In this equation  $Length\_REF$  is length of the reference tour and  $Length\_GRE$  is length of the greedy tour. Than bigger the value, the harder the game.

$$SSG = 1 - \frac{Length\_REF}{Length\_GRE} \quad (2)$$

This kind of tour complexity does not cover the ability to start from other goals than the lucky one. It represents how hard it is to calculate the reference tour starting from the lucky place. However, player cannot know the location of the lucky place before he/she starts the game and see where the goals are.

Assume that player can start the game from any place and will use *greedy* strategy. Game complexity is then defined as a ratio between the reference tour length and the average length of the greedy strategy applied for each goal as the start place. We use the equation (3) to calculate the game complexity. In this equation  $Length\_REF$  is length of the reference tour and  $Length\_GRE$  is an average length of the greedy tours.

$$Av = 1 - \frac{\overline{Length\_REF}}{\overline{Length\_GRE}} \quad (3)$$

That ratio represents how much the start location affects the distance which the player is expected to travel. However, this ratio does not present how ineffective the game could be if started from the worst location. It could be a situation when there is only one bad place to start a game, and it will not affect so much on the R value. Lower value means that the start location affects less to the game tour. The hardest game will have the highest value.

Tour complexity can also be calculated according to how much tour deviates from a straight line. For example, when travelling to the first goal, we have to choose the second goal and turn our moving towards that direction. We measure the angle (in degrees) how much player has to change the direction, and use this value as estimation for game complexity. We use the equation (4) to calculate the game complexity. In this equation  $\overline{angle}$  is the average change of angles. Lower value means that the tour is more similar to line than tour with higher value.

$$A = \frac{\overline{angle}}{180} \quad (4)$$

Next measure is *standard deviation* ( $\sigma$ ), which describes how far a set of numbers is spread out. A small value indicates that the data points tend to be very close to the average value. A high value indicates that the data points are very spread out around the average value. Using the previous greedy strategy to visit the goals, we calculate  $\sigma$  of all greedy tours to represent how many extra or less kilometers could be walked by player if he/she starts from the unlucky or the lucky start position effects on the final result[16]. This value represents that how far from the average the average tour of the greedy strategy is. Higher value means that there is higher change to have bad result and to finish the game with much more time than others.

Instead of counting how many meters the player walks, we count how many wrong choices he/she can make when following the greedy strategy. We use the order of the goals from the reference tour and count how many mismatch choices the tour from greedy strategy (Figure 42) has. This difference is calculated by starting from each goal as player can start game from any place. The value is then the average of the mismatch of these choices (MM).

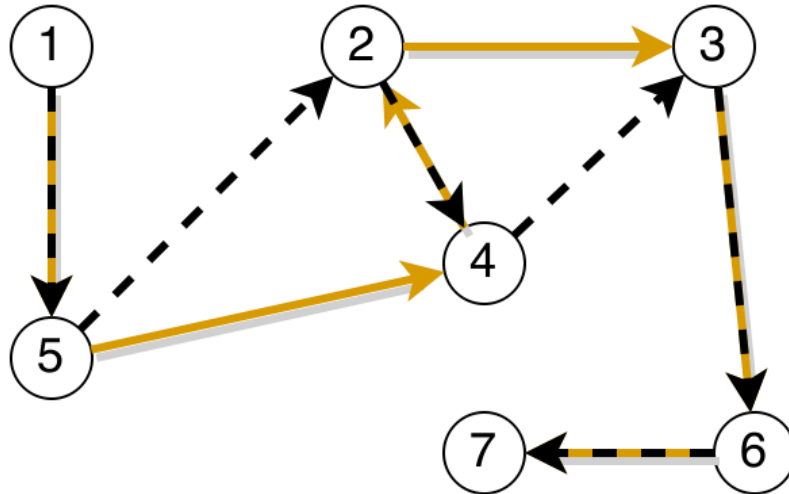


Figure 42: Difference between two tours. Mismatch value is  $MM=2$

We use the equation (5) to calculate the game complexity. In this equation  $t$  is tour,  $t_0$  is reference tour,  $f(t_0, t)$  is function calculates difference between two tours and  $\overline{f(t_0, t)}$  is the average value of all possible differences of the tours. Lower value means that the tour is more similar to the reference tour than tour with higher value.

$$MM = \overline{f(t_0, t)} \quad (5)$$

## 5.6 Reference tour by Tabu search algorithm

In this section, we describe how the reference tour is calculated. The problem is to find optimal order of the goals leading to shorted tour. This corresponds to *travelling salesman problem* (TSP) with the exceptions that player can freely choose the start place, and does not need to return back to it. The problem is NP-hard, meaning that there is no fast algorithm that guarantees to find optimal solution. We use here *Tabu search* algorithm as implemented in O-Mopsi.

Tabu search is a heuristic search method employing local search for solving optimization problems (Figure 43) The main purpose of the algorithm is not to stop in local optimum but jump from one local optimum to another in the hope of finding global optimum[17]. Main feature of the algorithm is a Tabu list of actions which are forbidden in the forthcoming iterations. The list is filled by previous solutions when finding local optimum and forbid neighborhood current solutions<sup>18</sup>.

---

<sup>18</sup> <http://www.redheur.org/sites/default/files/metodos/TS02.pdf>

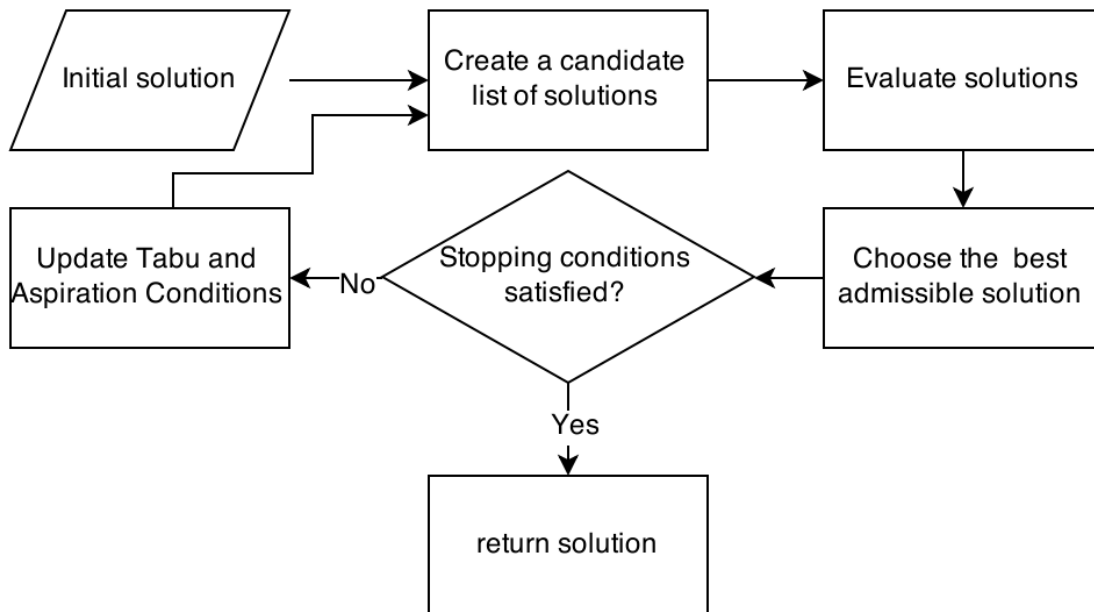


Figure 43: Tabu Search Algorithm

Tabu search for solving TSP we use the algorithm as follows<sup>19</sup>.

**TabuSearchTSP algorithm:**

- Generate initial solution
- Generate list of candidate solutions by swapping
- Calculate tour length of each new candidate
- Select the best candidate (with shortest distance)
- If the best candidate is not better than the previous best then stop
- Otherwise, repeat the generation steps until no further improvements found.

The initial solution is generated by greedy algorithm starting from the first goal. New candidate solutions are generated by swapping the order of two goals located next to each other in the tour. All pairs of goals are considered except those that are marked as *tabu*. The best candidate is then chosen. In case of ties, the first one in the list is chosen. The goals that were swapped, are added into the tabu list so that they cannot be involved in a swap in the future iterations. This process is demonstrated in Figure 44.

---

<sup>19</sup> <http://tonis.comli.com/index.php?page=readArticle&ald=9>

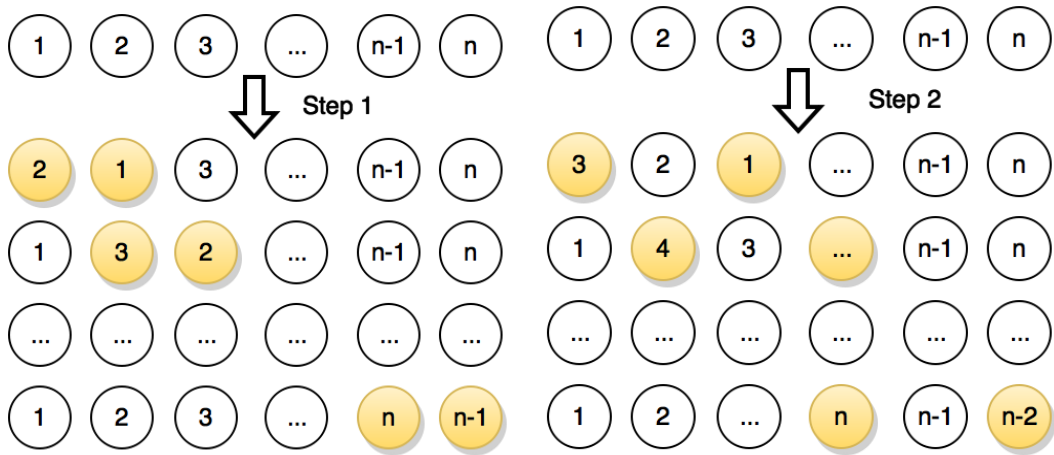


Figure 44: Tabu search swap

The best solution found is used as the initial solution for the next iteration. Tabu list can hold a maximum of  $k=n/2$  conditions, where  $n$  is the number of goals in the problem. In case  $n = \{1, 2, 3, 4, 5\}$   $k = 1$  always.

We consider example of the main steps of the algorithm. We have got a game with distance matrix as in Figure 45. For this example tabu list size is  $k=n/2=6/2=3$ , initial solution is  $t_0 = 1-5-6-4-3-2$  and  $l = 0.15 + 0.26 + 0.32 + 0.51 + 0.29 = 1.53$  (Figure 46).

$$\begin{pmatrix} 0 & 0.83 & 0.56 & 0.17 & 0.15 & 0.27 \\ 0.83 & 0 & 0.29 & 0.75 & 0.89 & 0.63 \\ 0.53 & 0.29 & 0 & 0.51 & 0.60 & 0.34 \\ 0.17 & 0.75 & 0.51 & 0 & 0.32 & 0.32 \\ 0.15 & 0.89 & 0.60 & 0.32 & 0 & 0.26 \\ 0.27 & 0.63 & 0.34 & 0.32 & 0.26 & 0 \end{pmatrix}$$

Figure 45 Distance matrix of the game

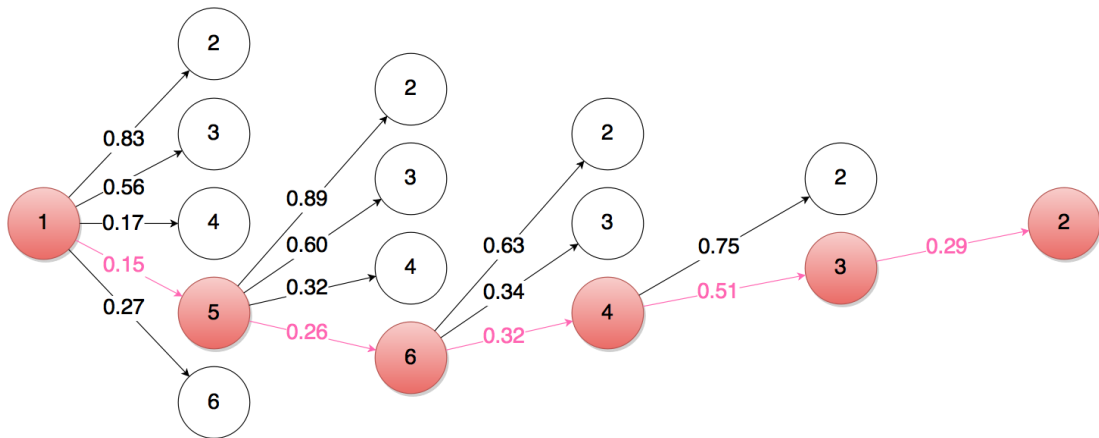


Figure 46 Steps of the greedy algorithm

The best candidate is  $t_1 = 6-5-1-4-3-2$ , it has tour length 1.39. The  $t_1$  was generated from  $t_0$  by swapping first and third goals in the tour. Tabu list updates and it holds  $\{(1,3)\}$  swapping indexes. The red font in the Table 2 highlights a forbidden solution because it uses swap action from tabu list. These solutions will not to be used to choose the best candidate. The best candidate is  $t_2 = 4-5-1-6-3-2$ , it has tour length 1.38. The  $t_2$  was generated from  $t_1$  by swapping first and fourth goals in the tour. Tabu list updates and it holds  $\{(1,4), (1,3)\}$  swapping indexes.

Table 2 List of candidate solutions

Iteration 1			Iteration 2		
step #	Solutions	Length of tour	step #	Solutions	Length of tour
1	1-5-6-4-3-2	1.53	1	6-5-1-4-3-2	1.39
	<b>5-1-6-4-3-2</b>	1.54		<b>5-6-1-4-3-2</b>	1.50
	1- <b>6-5</b> -4-3-2	1.65		6- <b>1-5</b> -3-4-2	2.28
	1-5- <b>4-6</b> -3-2	1.43		6-5- <b>4-1</b> -3-2	1.61
	1-5-6- <b>3-4</b> -2	2.02		6-5-1- <b>3-4</b> -2	2.24
	1-5-6-4- <b>2-3</b>	1.77		6-5-1-4- <b>2-3</b>	1.63
	2	1-5-6-4-3-2		1.53	2
<b>6-5-1</b> -4-3-2		1.39	<b>1-5-6-4-3-2</b>		
1- <b>4-6-5</b> -3-2		1.65	6- <b>4-1-5</b> -3-2	1.54	
1-5- <b>3-4-6</b> -2		2.22	6-5- <b>3-4-1</b> -2	2.38	
1-5-6- <b>2-3-4</b>		1.85	6-5-1- <b>2-3-4</b>	2.05	
3	1-5-6-4-3-2	1.53	3	6-5-1-4-3-2	1.39
	<b>4-5-6-1</b> -3-2	1.70		<b>4-5-1-6</b> -3-2	1.38
	1- <b>3-6-4-5</b> -2	2.43		6- <b>3-1-4-5</b> -2	2.29
	1-5- <b>2-4-3-6</b>	2.64		6-5- <b>2-4-3-1</b>	2.97
4	1-5-6-4-3-2	1.53	4	6-5-1-4-3-2	1.39
	<b>3-5-6-4-1</b> -2	2.19		<b>3-5-1-4-6</b> -2	1.88
	1- <b>2-6-4-3-5</b>	2.90		6- <b>2-1-4-3-5</b>	2.75
5	1-5-6-4-3-2	1.53	5	6-5-1-4-3-2	1.39
	<b>2-5-6-4-3-1</b>	2.54		<b>2-5-1-4-3-6</b>	2.07

Now, the best candidate is  $t_3 = 4-1-5-6-3-2$ , it has tour length 1.22. Tour  $t_3$  was generated from  $t_2$  by swapping second and third goals. Tabu list updates and it holds  $\{(2,3), (1,4), (1,3)\}$  swapping indexes (Table 3). On the fourth iteration, the algorithm stops and provides solution  $t_3 = 4-1-5-6-3-2$  (the tour with shortest distance).

Assuming the algorithm continues to work and the next swapping indexes are first and second then tabu list will be  $\{(1,2), (2,3), (1,4)\}$ . The algorithm has removed first added indexes because the size of the list is three.

Table 3 List of candidate solutions

Iteration 3			Iteration 4		
step #	Solutions	Length of tour	step #	Solutions	Length of tour
1	4-5-1-6-3-2	1.38	1	4-1-5-6-3-2	1.22
	<b>5-4-1-6-3-2</b>	1.40		<b>1-4-5-6-3-2</b>	1.39
	4- <b>1-5-6-3-2</b>	1.22		<b>4-5-1-6-3-2</b>	
	4-5- <b>6-1-3-2</b>	1.70		4-1- <b>6-5-3-2</b>	1.59
	4-5-1- <b>3-6-2</b>	2.01		4-1-5- <b>3-6-2</b>	1.91
	4-5-1-6- <b>2-3</b>	1.67		4-1-5-6- <b>2-3</b>	1.51
2	4-5-1-6-3-2	1.38	2	4-1-5-6-3-2	1.22
	<b>1-5-4-6-3-2</b>			<b>5-1-4-6-3-2</b>	
	4- <b>6-1-5-3-2</b>	1.63		4- <b>6-5-1-3-2</b>	1.59
	4-5- <b>3-6-1-2</b>	2.37		4-1- <b>3-6-5-2</b>	2.23
	4-5-1- <b>2-3-6</b>	1.94		4-1-5- <b>2-3-6</b>	1.85
3	4-5-1-6-3-2	1.38	3	4-1-5-6-3-2	1.22
	<b>6-5-1-4-3-2</b>			<b>6-1-5-4-3-2</b>	
	4- <b>3-1-6-5-2</b>	2.49		4- <b>3-5-6-1-2</b>	2.47
	4-5- <b>2-6-3-1</b>	2.75		4-1- <b>2-6-3-5</b>	2.80
4	4-5-1-6-3-2	1.38	4	4-1-5-6-3-2	1.22
	<b>3-5-1-6-4-2</b>	2.09		<b>3-1-5-6-4-2</b>	2.04
	4- <b>2-1-6-3-5</b>	2.79		4- <b>2-5-6-3-1</b>	2.80
5	4-5-1-6-3-2	1.38	5	4-1-5-6-3-2	1.22
	<b>2-5-1-6-3-4</b>	2.16		<b>2-1-5-6-3-4</b>	2.10

### 5.7 Greedy algorithm for travelling salesman problem

We present a greedy algorithm based on the optimal choice on each iteration step, with the hope of finding global optimum (Figure 47). Greedy algorithm for TSP<sup>20</sup> is described below [18]. The algorithm returns the list of ordered goals.

1. Choose start goal and mark it as visited
2. Choose the closest goal to it, and mark it as visited.
3. Repeat the Step 2 until all goals have been visited.

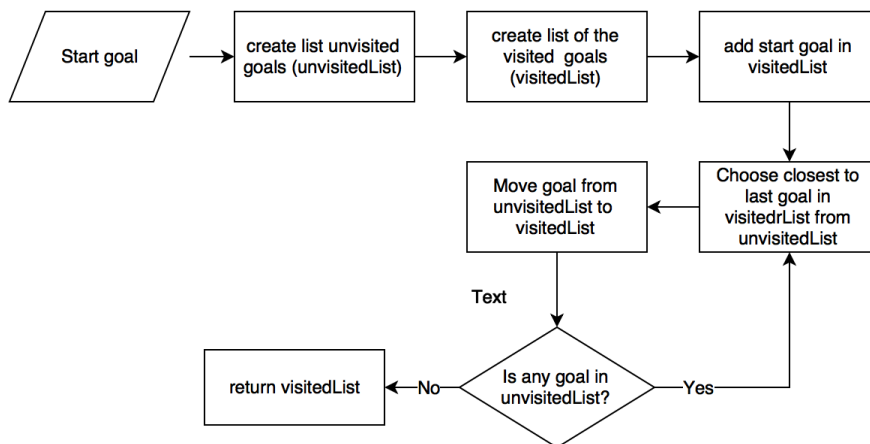


Figure 47: Greedy Algorithm

<sup>20</sup> <http://www.cs.usask.ca/classes/280/t2/Notes/travelingsalespoerson.pdf>

## 6. Experiments

We selected the most interesting existing O-Mopsi games, and collected the values of these complexity measures in Table 4. The games are sorted from the easiest to the hardest according to the *mismatch* (MM) criterion.

Table 4: Different values of game complexity

Game name	MM	SSG	Av	A	$\sigma$
Freeport town (easy)	0.50	0.00	0.24	0.06	1.46
Utransaarten U-käännös	0.80	0.00	0.02	0.26	0.09
Otsola	0.89	<0.01	0.24	0.36	0.26
SciFest 2015 Mini	1.00	<0.01	0.13	0.23	0.81
Areena Short Track 1	1.40	0.00	0.06	0.37	0.07
Kuntorastit 17.4.2014	1.83	0.00	0.13	0.16	0.13
SciFest_Medium	2.13	0.00	0.21	0.40	0.08
SciFest2013 Easy	2.50	0.03	0.11	0.55	0.37
Tallinn Long tour	2.50	0.00	0.24	0.22	0.05
Teuva (long)	2.60	0.00	0.33	0.28	0.03
Niinivaara 14.11.2014	3.60	0.07	0.14	0.45	1.05
Tour Lakeus	3.79	0.03	0.14	0.27	1.28
Symmetry	4.00	0.08	0.23	0.39	0.39
Savonlinna	4.46	0.16	0.11	0.34	2.03
Luxembourg Ville Haute	5.28	0.00	0.05	0.34	0.19

The SSG game complexity measure does not provide useful information. Almost all of the game has value 0, which means they are easy games to finish if the player started from the lucky place. One of the hardest games according to this is *Savonlinna* game (Figure 48). It is a good example of a game where player has to think which goal he/she should reach next. This game has also high  $\sigma$  value, which means the tour might be longer than average if started from unlucky location. The *angle* criterion has higher than average value and it means that to finish the game, the player has to make more turns as the targets are not located on a line. Also the mismatch criterion indicates that this is harder game. To sum up, all criteria indicate that this game is harder than average game. But for this game reference tour is not optimal and if player starts from the corner of the map he/she can easily make a round tour without crossing own path, so cannot be hard.



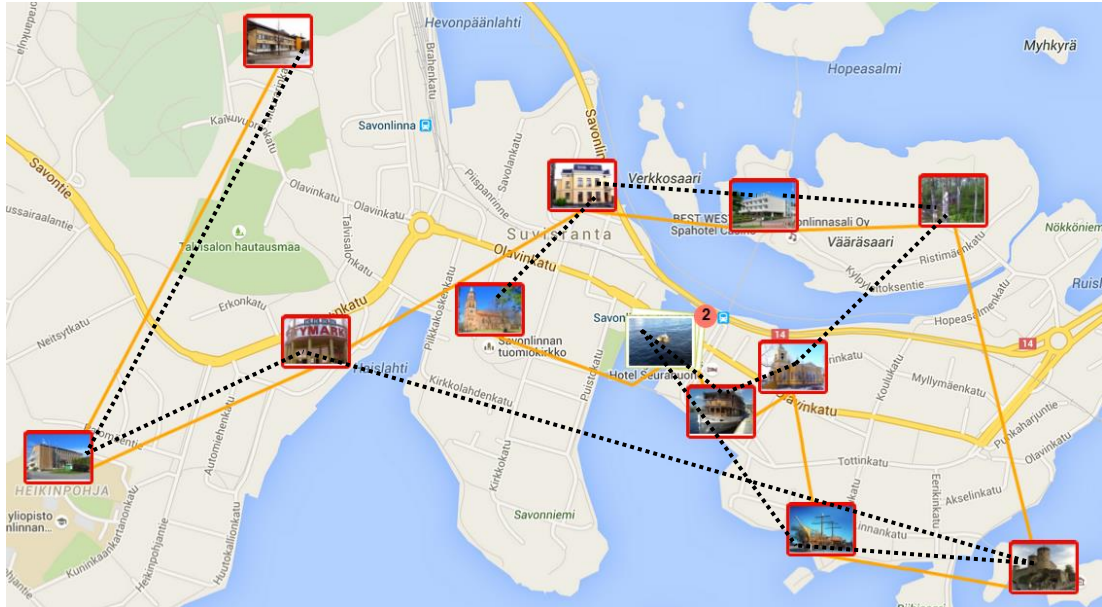


Figure 48: Savonlinna game

Games with short distance of the tour (full cluster of the goals) are not good for the angle criterion. A game considered hard game by this criterion have low average criterion and  $\sigma$ -value as the game (SciFest 2013 easy) has very low dependency on the start location even if it the tour required many turns along the way (Figure 49). This criterion is not good for games with small number of goals, as it will imply the game much harder than it really is. This criterion also does not provide correct results for player if he starts the tour from the wrong goal. However, the angle criteria can recognize easy games, such as Freeport, even with long distance if the game tour is close to strait line. The mismatch criterion recognizes the SciFest2013 Easy as medium game, but not as hard as the angle criteria. This is because it contains a cluster of the goals and the player can find a good tour with only 2.5 mismatches when starting from random location.

The standard deviation criterion recognizes *Tallinn Long tour* as an easy game, but average criterion considers it as a hard game (Figure 50). The game contains a big cluster of the goals in the middle but otherwise the layout is quite linear along west-east direction. Starting location has a big effect of the overall length of a tour. This is why  $\sigma$  gives big value according to average criterion. The mismatch criterion considers it only as a game of medium complexity. Player has to think where to go when he/she reaches the cluster of the goals to make optimal choice.



Figure 49: SciFest2013 Easy (left) Freeport town (easy) (right)

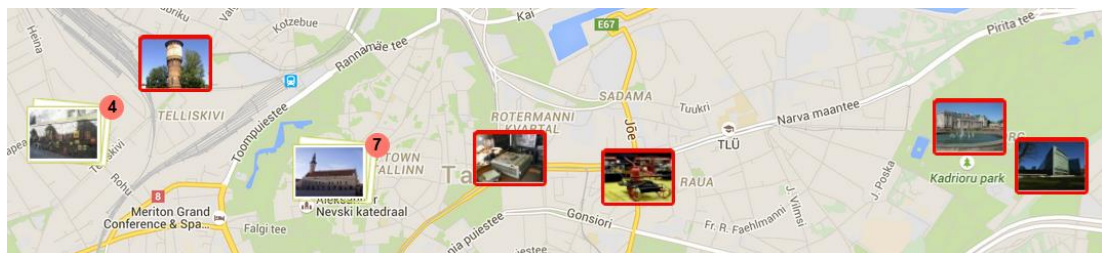


Figure 50: Tallinn Long tour

The mismatch criterion works well in case when the game contains many targets; it then estimates the game as a hard or medium. But in case those games have pattern like line it correctly recognizes that the game is easy. It does not depend on the length of the tour, which can also provide misleading result. Even when some choices can differ only by few meters, such game will be classified as hard even though the different orders make no big difference on the performance of the player.

## 7. Future work improving quality of the goals

When automatically selecting the goals, we take into account if player has visited them in the past. This happens when multiple games include the same goal and player has been active in the region. We downgrade goals that the player has already visited, by giving them less chance to appear as a part of another game. This means that the game goals will be new to the player, and therefore, playing is more interesting.

Sometimes player cannot finish a game because it has unreachable goals. We therefore upgrade goals that other players reach, only if that player finishes the game, and give them more chance to appear in other games.

All games that have been generated and finished by other players, those games have *published* status and have game master *Mopsi* (all requested games from mobile application have Mopsi as game master). Algorithm has to collect all games with this status, and create a list of them for the game master. If the list contains candidate goals that have the same information that goal should be moved to top of the rank.

To make down ranking for the goals we have to collect a list of all games that the player started to play. Those games are used in the same way as for up ranking. All possible goal candidates with the same pointID and file name have to be moved to the bottom of the rank list.

To balance the location of the goals on the map we recognize single goals located away from others. We could calculate the location of the *centroid* and check the distance from each of the goal. If more than half of the distances are bigger than the average distance to the centroid then we remove the farthest goals and add next one from the query.

The goals with a title could use algorithm similarity to the tags increase amount of the goals that relates somehow to the tags. It would be better to use than pure string matching between title and tag [19].

The game should not have goals far away from others. There should not be clusters of the goals which are too easily recognizable. The goals should be distributed on the map balanced to give the player a challenge to which goal should be visited first and which next.

## 8 Conclusions

In this thesis, we have presented a concept what is a good game for O-Mopsi location-based mobile orienteering game. We discuss the concept in detail, and give guidelines how to create a good game using O-Mopsi web page. Based on the concept and these guidelines, we introduce an algorithm to generate a good game automatically. The algorithm ranks the goals and evaluates the goodness of the game.

To generate better games, the algorithm is applied on the area with high density of distributed goals. In low density area, there might not be enough data to generate a game that fits to the given game parameters.

We also discussed methods how to improve the goal ranking of the goals further, and presented approach how to make more balanced goal distribution on the map. The proposed algorithm is brute force, which could be improved by replacing it by heuristic method to pick the goals more efficiently. The system is based on tags. Other choices would be to use algorithm measuring word similarity instead of exact matching.

The game generation feature has been implemented as new module in the O-Mopsi web site. The windows phone software supports this module by allowing players to request automatic generated games. Overall, the thesis can be used for better understanding for the game generation module in O-Mopsi.

## References

- [1] Baer, Ralph H. (Manchester, NH), Rusch, William T. (Hollis, NH), Harrison, William L. (Nashua, NH), TELEVISION GAMING APPARATUS AND METHOD, 1972.
- [2] M. N. K Boulos and. S.P. Yang, "Exergames for health and fitness: the toles of GPS and geosocial apps", *International Journal of Health Geographics*, 2013.
- [3] W. Wiegmans, Location-based gaming, Enschede: Bachelor thesis Telematics, University of Twente, 2005.
- [4] L. Meng, Z. Liqiu, *Map-based Mobile Services: Design, Interaction and Usability*, Springer, Berlin Heidelberg, 2008.
- [5] I.R. Larissa Hjorth, *Gaming in Social, Locative and Mobile Media*, Palgrave Macmillan, 2014.
- [6] National Academy of Public Administration, Commission on Engineering and Technical Systems; National Research Council (U.S.). Committee on the Future of the Global Positioning System, *The Global Positioning System: A Shared National Asset*, National Academies Press, 1995.
- [7] M. Saile, J. Hense, H. Mandl and M. Klevers, "Psychological Perspectives on Motivation through Gamification," *Interaction Design and Architecture(s)*, vol. 1, pp. 28-37, 2013.
- [8] D.R. Marins, Marcelo de O.D.Justo, Bernardo de A.M. Chaves, "SmartRabbit: A Mobile Exergame Using Geolocation," *Proceedings of SBGames*, Salvador, 2011.
- [9] L. Valente, B. Feijó, "A Survey on Pervasive Mobile Games," *Monografias em Ciência da Computação*, July 2013.
- [10] A. Tabarcea, Z. Wan, K. Waga and P. Fränti, "O-Mopsi: Mobile Orienteering Game using Geotagged Photo," *Int. Conf. on Web Information Systems and Technologies (WebIST)*, Aachen, 2013.
- [11] Z. Wan, "O-Mopsi: Location-based Orienteering Mobile Game", M.Sc. thesis, University of Eastern Finland, 2014.

- [12] Snavely, N., Seitz, S. M., and Szeliski, R., "Photo tourism: exploring photo collections," *ACM Trans. Graph*, vol. 3, p. 835–846, 2002.
- [13] P. Fränti, J. Chen and A. Tabarcea, "Four Aspects of Relevance in Sharing Location-based Media: Content, Time, Location and Network.," *Int. Conf. on Web Information Systems and Technologies*, Noordwijkerhout, The Netherlands, 2011.
- [14] P. Fränti, J. Kuittinen, A. Tabarcea, L. Sakala, "MOPSI Location-based Search Engine: Concept, Architecture and Prototype," in *ACM Symposium on Applied Computing (SAC)*, Sierre, Switzerland, 2010.
- [15] C. Ardito, C. Sintoris, D. Raptis, N. Yiannoutsou, N. Avouris and M.F. Costabile, "Design guidelines for location-based mobile games for learning," *Social Applications for Lifelong Learning*, pp. 96-100, 2010.
- [16] M. George W Brown, "Standard deviation, standard error," *American Journal of Diseases of Children*, vol. 136, pp. 937-941, 1982.
- [17] D.T. Pham and D. Karaboga, *Intelligent Optimisation Techniques - Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*, London: Springer-Verlag, 2000.
- [18] Cormen, Leiserson, Rivest and Stein, *Introduction to Algorithms Third Edition*, Chapter 16 "Greedy Algorithms", London: The MIT Press, 2009.
- [19] M. Thangaraj, V. Gayathri , "A New Context Oriented Synonym Based Searching Technique for Digital Collection," *International Journal of Machine Learning and Computing*, vol. 1, pp. 100-103, 2011.

## Appendix 1: Code Usage

JavaScript example to request game from O-Mopsi server. It has two parts. First part is information to the server about what to do and second part is parameters that will be handle by module.

Parameters variable is JavaScript object that has properties: *latitude*, *longitude*, *tour\_length*, *goal\_distance*, *amount\_of\_goals* and *tags\_string*. The *latitude* and *longitude* is the location of the center of the bounding box for expected game area, *tour\_length* is expected distance of the reference tour, *goal\_distance* is average expected distance between goals, *goal\_distance* is excepted amount of goals, *tags\_string* is a string of the tags, words divided by comma or space symbol.

```
Object { latitude: 62.6001077874621,  
          longitude: 29.754701455776512,  
          tour_length: 4,  
          goal_distance: 300,  
          amount_of_goals: 6,  
          tags_string: "" }
```

The request should be send to the *OMopsiGameController.php* using properties *request\_type*, *userId* and *param*. Param it is a parameters variable.

```
"/o-mopsi/controller/OMopsiGameController.php"  
{ request_type: "generate_new_game",  
  userId: g_loggedInUserId,  
  param: JSON.stringify(param) }
```

The response contains properties: *goals*, *latlng* and *mes*. Goals and *latlng* is array variables and *mes* is a text variable.

```
Object { goals: Array[6],  
          latlng: Array[6],  
          mes: "game length 2.080470632091717 " }
```

The goal object contains *lat*, *lon*, *photoid*, *address*, *goalName*, *desc*, *goalId*, *photoURL*, *ItHasTitle*, *NearestRoadDistance*, *NearestRoadPoint* and *locSource*. Lat and lon it is geolocation coordinate of the goal, *photoid* it is Id of the photo in database, *address* is the address of the location of the goal, *goalName* and *desc* is name and description of the goal, *goalId* is a Id of the goal in the game database, *photoURL* is a link to the picture of the goal, *ItHasTitle* is marker of the quality of the title of the goal, *NearestRoadDistance* is distance to the nearest point on the road, *NearestRoadPoint* is object that contains latitude and longitude of the point on the road and *locSource* is information about source that provides location of the goal.

```
Object { lat: "62.595773935318",  
          lon: "29.7514754533768",  
          photoid: "190913_00-56-42_1181047380.jpg",  
          address: "",  
          goalName: "happy",  
          desc: "happy",
```

```

        goalId: "25441",
        photoURL:
        "http://cs.uef.fi/paikka/mobile_photo/190913_00-56-
        42_1181047380.jpg",
        ItHasTitle: "true",
        NearestRoadDistance: 1.5564574880299,
        NearestRoadPoint: Object ,
        locSource: "fixed"}

```

The name of the module is *OMopsiGenerateGameHandler.class.php*. It responses for generate game. We have to call function *generateNewGame* with parameters as was mentuated before. Parameters must be in JSON format.The *generateNewGameForMobile* has the same parameters but this function also save the game to the game list to the mopsi game master.

```

function generateNewGame($params)
function generateNewGameForMobile($params)

```

The module *GameComplexity.class.php* contains function *GameComplexity* and *SetNewGame*, the first one uses for initialisation for the class and second one for recalculate new game. The *game* variable it is array with *keys* latitude and longitude. Response is the float value.

```

function GameComplexity($game)
function SetNewGame($game)

```

The module *SortGoalsByQuality.class.php* evaluates and sorts goals by the quality. The *goals* variable contains four arrays: *name* of the goal, *sourceLoc* is location source of the goal and *location* of the goal. The response is the same object but in sorter order.

```

function SortGoalsByQuality($goals)
    Array ( [name] => Array (
        [sourceLoc] => Array (
        [location] => Array ( )
    )
    )

```

The module *GroupByGrid.class.php* creates the array of the centroids with linked to this array of the goals. The *goals* variable contains array of the *locations* the goals.

```

function GroupGoalsByGrid($goals)

```