A Mobile Location-Based Game Framework

João Tiago Pinheiro Neto Jacob[1],

[1] Faculdade de Engenharia da Universidade do Porto
joao.jacob@fe.up.pt

Abstract. As mobile platforms evolve, so does the concept of video games for these platforms. Location-based games are a recent type of games that explore the unique capabilities of the GPS-enabled mobile devices. However, the development of these games isn't easy as they often encounter several issues regarding the usage/availability of location-related content. This paper presents an in-depth analysis of these games and their issues and presents a solution to these problems through the creation of a framework. This framework also offers many features useful for location-based games, such as remote-map access, weather location and easy access to the device's GPS module. A proof-of-concept location-based game based upon said framework is also presented, successfully validating the model.

Keywords: Location-based games, mobile computing, GPS, location-based services, mobile games

1 Introduction

Location-based games are relatively new in the entertainment industry, as the first location-based games only came to be in 2002 [13]. These games are known for their unpredictability as they rely upon the user's real location (or other location data), as a means of input, or as a means of generating/accessing game specific content. As such, these games provide players with distinct gaming experiences, not only from player to player but also from location to location, effectively increasing the game's longevity and its possibilities. As these games use location services to function (usually a GPS - Global Positioning System - module for user positioning), they can almost exclusively be found on mobile platforms, as these are more prone to be the most adequate.

However, these games often become unplayable in situations where the location services are inoperative, such as inside buildings when playing a game that requires the usage of the GPS module, as the GPS needs line-of-sight with the sky in order to pinpoint the user's position. Still, some of these game's issues aren't so closely related with the hardware limitation, be it performance or communication wise. In fact, most of the issues found on these games are related to their location-based characteristics. So, while a game may be played adequately in the location A, it may be too difficult (or unplayable altogether) in the location B. Additionally, some locations may be more easily and naturally incorporated in the game's mechanics,

while some won't, usually due to the lack or poor quality of information that location has associated or generated within the game.

During the analysis of some location-based games and location-based game's platforms, some generic issues were identified. With the goal of solving most of these issues, a location-based game framework was designed and implemented. Finally, a location-based game was developed atop said framework [18] in order to serve as a proof-of-concept and determine if the designed framework was able to cope with the issues. The testing of the game (Geo-Wars) is being made by several anonymous users, but the feedback gathered for now helped to fine tune the framework, and only some minor issues are amiss.

This paper accommodates a small section describing some location-based games and their technology as well as a "location-based games' issues" section with an in-depth analysis of the found issues. A section detailing both the framework and its implementation and another section regarding the concept of the location-based game Geo Wars is also present. Finally, the discussion and the conclusions and future work sections, portraying a critical and objective analysis of the developed work and its result, follows.

2 Location Based Games

A Location-based Game is a game that uses the player's physical location (or any other location) as a means of input or to generate or access location-based information. These games are almost exclusively available on mobile platforms [16]

Location-based games haven't been around for too long. In fact, they would only see any commercial use in 2002 with the arrival of Botfighters [2], the first pay-per-locate GPS game. The concept was simple: each player was a robot with the mission of destroying the enemy robots. In order to play the game, the player would move around the city, scanning for enemy robots, which would be other players with the same goal.

With the birth of this genre only roughly eight years ago, location-based games have now gained a considerable popularity. For an instance, Geocaching [6], probably the most popular of geocaching games, claims to have more than a million registered users. The idea behind it is even simpler: a player stashes a "cache" (with contents in accordance with the geocaching's rules) anywhere in the world and shares its approximate location with other geocachers around the globe. Other players will attempt to discover this cache by solving some riddles and exploring the general area the cache is known to be.

Of course other games, although using the physical location of the player, need to be played with a mobile location-capable device. Pac-Manhattan [12] is such a game. In it, players will have to enact a classic Pac-Man game. In order to do so, ten players are required: one to be Pac-Man, four other to be ghosts, and the remaining five to control and coordinate the actions of the others via mobile phone. As the Pac-Man player wanders around the streets of a real city, the player responsible for controlling him will guide him through the streets, avoiding the ghosts and, of course, eating pellets.

As location-based games gained popularity, several platforms of mobile games emerged, with two having a particular success. One of them is Groundspeak [5], the platform that contains not only the Geocaching game, among others, but also the Whereigo tool, a tool for creating and playing GPS-enabled games [11]. The other, Locomatrix [4], currently in quick expansion, offers two games: Treasure hunt, a geocaching like location based game, with a more virtual component, as the player has to use pictures to solve riddles and find places, and fruit farmer, a game where the player must collect "fruits" around him by moving around in the real world, avoiding also having these fruits stolen by other farmers. As it can be seen, location-based games have come to be quite a success from their origins in the early 2002.

2.1 Location-Based Games' Issues

The issues that were found in most location-based games usually fall into one of these following categories:

- Game design issues
- Hardware limitations
- Location-related information availability and suitability
- Player's fitness and pace

There are several possible solutions for these issues.

2.1.1 Game design

Regarding game design issues it is important to keep in mind that location-based games involve player exposure to physical interaction with the real world. Meaning that it is important to consider where and how the player will attempt to play the game and try to keep the player safe. Being a location-based game it always involves some degree of unpredictability. However, that unpredictability can be reduced if the player's behavior is limited thanks to the gameplay. For an instance, setting a racing location-based game that involves the player to go from A to B may be tempting for some player to attempt to break the record using a motorized vehicle and thus risking his safety and that of those that surround him. However, if a restraint is set, by determining another objective like, considering the previous example, stating that the player's top speed must not exceed 30Km/h, the risk of a player speeding his way through the game is reduced. Additionally, and in order to ensure that the game is played on foot, using the devices' sensors such as the accelerometer to work as a pedometer may help. These solutions don't alter the game altogether but help ensure that the gaming experience remains constant and that the game's unpredictability is lowered. Since game design is such a vast area, it is difficult to analyze location-based game design issues without going through case by case analysis. Even so, the important thing is to keep in mind what possible issues may arise from the conceptualized gameplay.

2.1.2 Hardware limitations

Hardware limitations are difficult to overcome. In regular mobile games, the developer might limit the user's actions or the game's requirements. However, in the

case of location-based games, since other hardware is often used, it is mandatory to have them taken into account.

Since these games often rely on GPS or data connections in order to be playable it is often the case where the player, unable to have one or neither of these services available, is incapable of playing the game. However, if the game's mechanics allowed playing the game without using GPS, but still using some location as an input (such as from logs or via direct input by the user) this situation could be avoided, sacrificing gameplay experience. On the other hand, the game could be played indoors or without any preparation whatsoever, which would be sure to please the most casua gamers. Regarding data connections, more specifically via mobile carier, since these are usually paid-for connection and have a limited coverage, the most adventurous of the players will probably find himself in a situation where the access to this service is precarious or too expensive. Under these circumstances, allowing the player to still play the game, either using cached location-related data to be used or, in case that location-specific data is lacking, using other location data. Such approach would allow the player to overcome this issue and still play a location-based game. In the case of the game using location data that isn't specific for the location of where the game is being played, the game can still use the GPS signal to move the player's avatar in the game (if applicable). Obviously, although mapping his movement in the game, the game itself won't be considering the players true surroundings. If the game requires some data to be transferred, such as scores, statistics or saved games, this data transfer can often be postponed with no harm to the game.

2.1.3 Location-related information availability and suitability

Since location-based games often use maps, weather information, or any other kind of location-related information in order to make the game truly unique and location-based, it is not uncommon to find location-based games that are rendered unplayable in many parts of the globe, due to requiring information that is either not available or that isn't relevant and cannot be used in the game. Creating such a game, like Pac-Manhattan [12], means that the game is unplayable outside of the area it was designed to be played, and thus, has a narrowed-down target audience. There are three possible solutions for this issue: not using location-related information, generating the needed information randomly, by using the player's GPS position as an input, so that, even though the location's true novelty is lost, a new virtual novelty is created for that location, guaranteeing that the game is playable worldwide. Alternatively, and since mobile devices storage capabilities are now into the tens of gigabytes, it is possible to store data for all/several of the needed locations. However, and while this may ensure that the novelty of the location is used (even if still not available for some people), that information is prone to become out-dated, particularly if that information is very ephemeral, such as that of weather or traffic. Ultimately, the usage of remotely stored location-related content that is often updated via web-services will guarantee that the needed information is up-to-date and available to everyone without sacrificing the device's limited storage. Unfortunately, this implies the usage of data connections, limited location coverage (as the information may not still be available for every playable location), and possibly remotely inaccessible servers. Often, even if the

location-related content is available, it may still not be of any use if the game's mechanics doesn't take it into account.

2.1.4 Player's fitness and pace

As it happens with most location-based games, the player's movement around the real world is used as an input, often to move his avatar around the virtual world. And like some games, such as Fruit Farmer [4], the player's speed and stamina is determinant for the outcome of the game. Unfortunately, this means that many of such location-based games are too difficult or altogether unplayable by the unfit player, providing an unbalanced gaming experience. If, however, the game was able to take the players pace into account, balancing its difficulty in real-time, the player would feel that the game was suited to him, and was still challenging without being too easy or overwhelmingly difficult. In the case of the "Zombies, Run" [14] game, (a game where the player must go from point A to point B in a real map by physically moving from the real point A to real point B while avoiding the hordes of Zombies that will pursue him) if it automatically adjusted the number of zombies and their speed to the pace of the player (taking into account the distance travelled, the average speed and the current speed), the game would provide a custom-tailored experience. Of course the game would still value the fastest of the players via metagaming (such as scores or boards), but the game would be playable for everyone. Something often overlooked by location based games is that the player will probably need to stop to catch his breath. If possible (with the exception of real-time multiplayer location-based games) the game should be paused automatically whenever the player's stopped, or , if not pausing, slowing the game down significantly, giving the player the chance to gather his strength, as opposed to him losing the game due to pausing for a few seconds.

3 Location-Based Game Framework

In order to solve the aforementioned found issues (using some of the possible solutions for them), a generic solution for a location-based game, a location-based game framework, was designed. This solution, aims to be as general as possible, so as to be usable by any location-based game that requires support for the previously numbered issues.

The implementation of both the game's concept and the core components of the framework followed a "waterfall" methodology variant called the Sashimi model [20], allowing to jump back and forth between the several phases of development, such as the game's implementation and the game's design as well as the distinct phases of the framework development, so as to use the feedback each of them provided to alter and complete other phases. After the creation of a first fully working prototype based on this framework, this product was tested on the field both by the developer, some other volunteers and also by the community at xda-developers.com. This evaluation allowed the adoption of the "spiral" model, a model that uses user feedback to reevaluate product features and the whole design of the solution, improving it with each iteration. Since both (Sashimi and spiral models) are agile

methodologies they were sure to provide flexibility in order to spot and solve problems on the fly, even if spotted during distinct phases of the development [1]. The design here presented for both the framework and the location based game prototype (Geo Wars) is the final one, considering already most of the provided feedback. The design of the solution shows the relevant key elements needed for it to function.

Each step of the methodology and the design of the components of the framework will be covered in further detail during the sub-sections that follow.

3.1 Location-Based Game Framework Architecture

The created framework is capable of accessing either local (on the mobile device) or remote (stored in a server) location-specific content. Meaning that the player can play a cached game, one that he has already played or has already downloaded the needed content, or the player can easily play a new, never before played game. This allows the player to:

- Play a game even without data connections, simply playing a saved game,
- Playing a new game, without worrying about having the necessary data to play it.

Furthermore the framework supports the option of letting the player decide to use his GPS to play the game, or not. Obviously, for some games, this may or may not make sense; e.g.: playing a Manhattan Pac-Man with no GPS at all means that the player will be playing a regular pac-man game. Still, it is usually better than no game, or an unplayable game, at all.

If the player decides to not use his GPS (or the service is unavailable), the location information can be either inserted into the system via prompting the user (e.g.: selecting the country or city he is playing at), or allowing the input to be done another way; e.g.: if the player's avatar is moved by mapping the player's real position, obtained via GPS, with the avatar's virtual position, then the avatar may also be played with on-screen controls, even though the game may lose its location-based characteristics. So, regarding the GPS a player can decide to:

- Turn the GPS off, while giving the needed data for the game to remain playable,
- Keep the GPS on, in order to take full advantage of the location-based characteristics of the game.

Additionally, the player may opt to create and share a customized location-based game's settings online. This allows for the games to gain a social and competitive aspect, as players may play against other players (through a game specific scoreboard), and can also share and check ideas for location-based game's settings. This is important, as some players will be unable to play a desired location-based game. For instance, a player may wish to play a game as if he was in Paris. Even though the player has GPS available, he isn't there. So, the option of playing a game with content that isn't available for the player's location while using the GPS enabled device to sense the player's movement around is possible.

The framework's architecture can be summarized as follows:

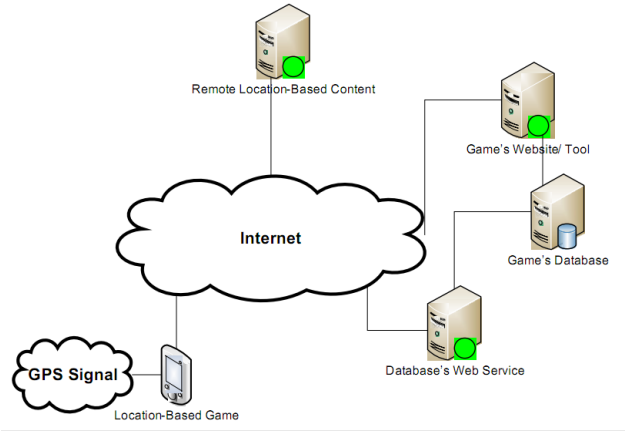

Fig 1: The solution's design

The Figure 1 summarizes some of these capabilities of the framework, in a physical architecture point of view of the solution. The framework supports the usage of a database's web service that allows for accessing games' scores and loading previously created games (using the game's website/tool and the game's database for storing this content). It is also explicit the access of remote location-based content via the internet (when needed), while the location-based game client itself runs on the device.

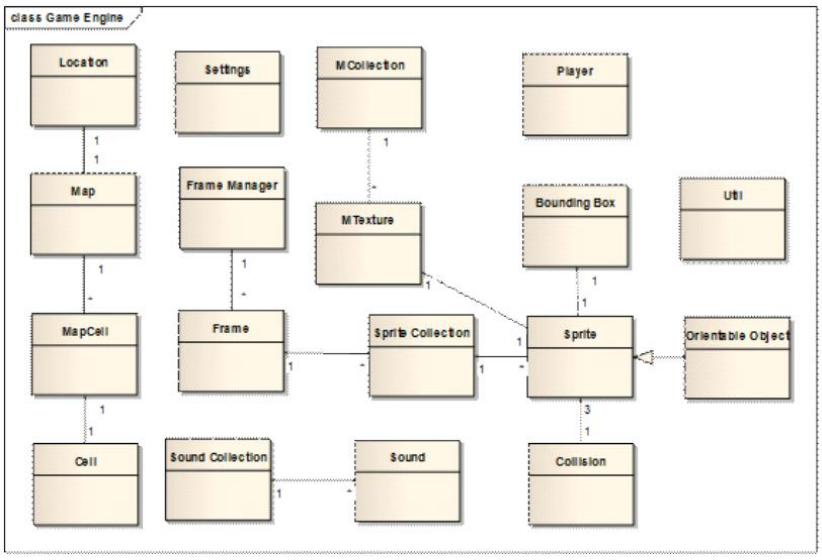

Fig. 2: Diagram depicting the core components of the framework

While some of the components presented in the Figure 2 provide features present in most game development specific frameworks, such as the Sprite class, Bounding

Box based collisions, Frames and Frame Managers, as well as sound playback, and some generic functions, present in the Util class, such as cache for sine/cosine operations, it also offers some distinct ones that will be here described. Other functionalities such as texture caching, sprite animation and transformation won't be explained, as they are common in most game development frameworks and aren't both the scope of this paper nor the most valuable assets of this project.

Cell: This structure holds information regarding the type of cell, the considered atomic structure type of a map. The type of a Cell is an enumeration and can be Building, Grass, Water or Road. It can be expanded to include other types of unitary cell blocks or points of interest as needed.

MapCell: A MapCell is the smallest, atomic element that constitutes a Map. It holds information of a particular part of a map, such as the type of that part (the Cell) and the coordinates of it.

Map: A Map is formed by several MapCells, detailing the maps content. Here is an image depicting the information present in a Google Maps Static API image versus the respective generated Map structure containing the location-based information found on the original map:

Fig. 3: Original Google map and visual representation of the in-game generated map

As it can be seen in Figure 3 a Map will hold information regarding streets among other elements (such as green areas, buildings, etc). Its resolution depends on the number of MapCells available for storing that information. Additionally, a Map will also contain a simple implementation of the A* algorithm with 8-conectivity and Manhattan distance heuristic in order to calculate shortest paths along roads or any other type/types of MapCell [21]. If the start and/or finishing nodes given to the A* algorithm are not accessible nodes, the algorithm will replace them with the closest node possible in order to function fully.

Location: The Location class is responsible for accessing all location-based services and other location related content. It will access a Google Maps Static API image with the given address or, in case an address wasn't given, it will use the built in GPS module of the device in order to determine the player's position and so, get the respective image. Afterwards, it will filter the image, storing a Map representing the information found (such as roads, buildings, etc). It is also responsible for

constantly updating the Player's position in accord with the Map (so that the virtual positioning of the Player matches the real positioning of the player) and also accesses Google's Geo Referencing Web Service to determine the city and country the player is at in order to access the weather web service and so, download the local weather and store it. Other features can be extended, or created in order to fulfill game-specific needs.

As expected from a framework, each of these features may or may not be implemented or used in the location-based game. Furthermore, some classes may be extended or implemented providing the necessary flexibility that any game requires from such a tool. The framework currently provides features that are most common in location-based games, but even a game that needs little to none of the features provided will at least find the usage of the GPS module, accelerometer, web services, simplified.

4 Location-Based Game: Geo Wars

In order to both test and prove the framework's capabilities, a location-based game was created on top of it.

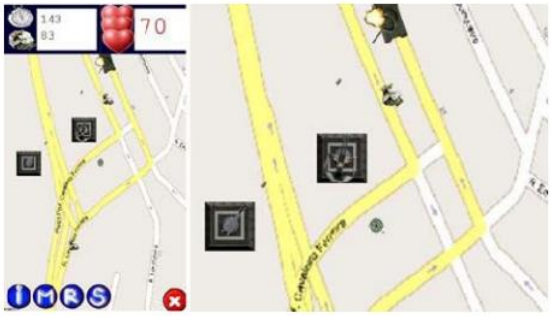

Fig. 4: Geo Wars look and feel

The game, Geo Wars, is a free to play location-based tower-defense game. In it, the player takes the role of a general with the goal of defending his sector (a portion of a map) from enemy forces, depicted in Figure 4.

The player must resist several waves of enemies, either by physically moving around, evading enemy fire or luring them into friendly crossfire situations, or by building defensive towers, each with its unique strengths and weaknesses. The enemy attacks by land, air and sea, using tanks, soldiers, airplanes and cruisers. While tanks and soldiers can only move around streets or parks, aircrafts can move anywhere in the map. Each unit has its own firepower, primary and secondary objectives (some units may prefer targeting specific towers rather than the general) and A.I.. Player uses money to build towers, money that can be gained over time, by destroying enemy units, or by physically moving to the locations of virtual bags of money displayed on screen. The game loads saved games settings via a web-service that

accesses a remote database that contains player-related saved games created using the online portal.

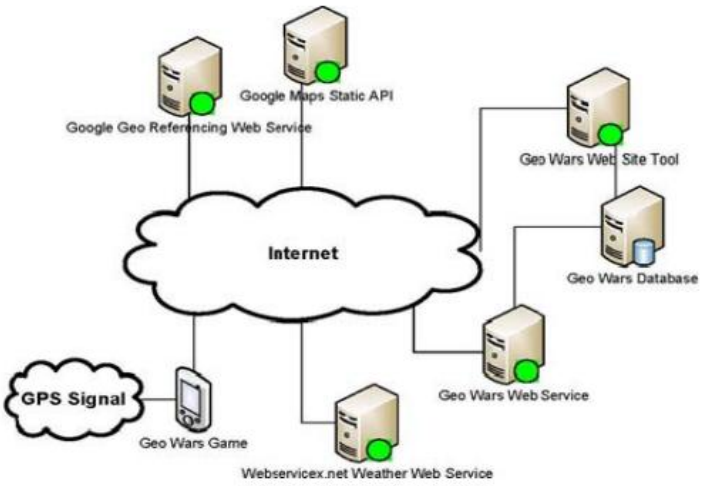

Fig. 5: Geo-Wars Physical Architecture

As far as location-based content, the game uses Google's Geo-Referencing webservice to determine the current city the player is at (in order to download weather data for it), Google map's static API to access the map of the game and Webservicex.net Weather webservice to determine the weather for any given city, as shown in Figure 5.

The game also takes into account weather (increasing the enemies speed if the weather is sunny and slowing it if it's cloudy or rainy), and allows the game to be played indoors by specifying an address for the game to be played or a cached map and by disabling the general's movement (which can make the game a bit harder to complete).

The full description of the game's architecture, features and a more thorough analysis of the users' feedback can be found here [18].

5 Discussion

Regarding the used features of the location-based framework by the Geo Wars location-based game, and comparing them with the code that would be needed to write if the game wasn't based atop said framework, for an instance, using only the directx mobile framework, the gps intermediate driver among other available solutions, it is reasonable to conclude that the framework, for this proof-of-concept game, allowed a faster implementation of the game's concept. In some situations where the framework was too generic for the game, e.g.: when writing the enemies' AI, it still provided some needed features (such as picking/collision detection) with a simple extension of the class Sprite or OrientableObject. Other features that are

probably needed in many location-based games, such as downloading maps, looking up addresses or accessing the player's last coordinates are easy enough to use and extra effort was put into making these features easily available.

Of course, judging from the fact that the only game developed so far using this framework was Geo Wars, it is difficult to determine what other features might this solution be lacking. Still, Geo Wars seemed to be well received by the gaming community [19], and although it would be incorrect to assume that any good game comes necessarily from a good framework, it is important to note that Geo Wars itself was developed in a very short time, thanks to the framework.

However, initially, the game was somewhat simpler, as the player could not choose where to play the game. Fortunately, the framework already allowed the download of a map given a coordinate, so it was only a matter of adding the ability of converting an address to a coordinate, and using said coordinate to play the game.

Performance wise, the framework is capable of a very modest performance, mainly due to the fact of relying on directx mobile for its graphics. Since the directx mobile framework isn't as used as its desktop counterpart, some OEM (Original Equipment Manufacturer) don't optimize their video drivers for directx mobile. However, Geo Wars is still capable of being run at about 20 frames per second in high-end mobile devices.

6 Conclusions and Future Work

During the course of this project, it became notorious that some limitations in location-based games were difficult if not impossible to overcome completely due to their natural unpredictability. As such, some of the mentioned issues were not solved as much as they were avoided. An example of this is the lack of GPS coverage in buildings that was overcome with the feature of allowing the player to play a game that is not based on his location. This doesn't constitute a solution for the problem but an alternative to not being able to play a location-based game at all. However, and thanks to the methodology based on the agile Sashimi model, that allowed to adjust requirements, features and the design of the solution even as other phases of the project went on being developed, many solutions for issues of location-based games that were only spotted during the implementation phase of the project were included in the game's design. Additionally thanks to the creation of a general location-based game solution (the previously mentioned framework, the base upon Geo Wars was implemented), the game's final concept was only elaborated and completed long after a simpler and different location based game was implemented using the very same general design, proving that the generic approach to the creation of location-based games was possible through the said framework.

The game Geo Wars has had a good acceptance in the mobile gaming community as it can be seen in this xda-developers thread [19] used for testing and gathering of feedback to be used in subsequent iterations. So much that the framework is being ported to 3D so that Geo Wars may also be a 3D game. Prototype versions for both 2D and 3D are also available for download in said thread as well as a small video depicting a Geo Wars match. Many of the testers of the game asked for a multiplayer

version of the game. Alas, the framework, as of now, is incapable of allowing such a feature, although it will most likely be altered in order to accommodate that possibility. Additionally, due to the fact of newer, more powerful devices and mobile OS are now surfacing, extra effort is being put in recreating the current framework in order to support Android OS.

References

1. Hunicke ,Robin, LeBlanc ,Marc, Zubek ,Robert , "MDA: A Formal Approach to Game Design and Game Research "
2. Novak, Jeannie (2008), "Game Development Essentials", Delmar Cengage Learning
3. Steiniger, Stefan, Neun, Moritz Edwardes, Alistair (2006) "Foundations of Location Based Services" in "Lecture Notes on LBS", 2006.
4. Locomatrix, 2010. http://www.locomatrix.com/about.html , 10/2009.
5. Groundspeak 2010 http://www.groundspeak.com/ , 10/2009.
6. GroundSpeak.Geocaching, 2010.http://www.geocaching.com/ , 10/2009.
7. GPSgames.org.GPSGames.org, 2009 . http://www.gpsgames.org/ , 10/2009.
8. GPS Games. Minute War, 2006. http://ultimategps.com/gps_fun.html , 10/2009.
9. GPS Games. Shutterspot http://www.gpsgames.org/index.php?option=com_wrapper&wrap=Shutterspot , 10/2009.
10. Navigadget. "Take down your targets! A gps phone tag game." http://www.navigadget.com/index.php/2006/04/04/take-down-your-targets-a-gps-phone-tag-game , 10/2009
11. GroundSpeak, WhereIGo, 2008. http://www.wherigo.com/, 10/2009
12. Pac Manhattan http://www.pacmanhattan.com/about.php, 10/2009
13. Dodson, Sean (2002) "Ready, aim, text", The Guardian, http://www.guardian.co.uk/technology/2002/aug/15/electronicgoods.games, 5/2010
14. Fikkert, Erik (2008), "Zombies, Run" http://www.androidapps.com/t/zombies-run, 9/2010
15. Araújo, Manuel, Roque, Licínio (2009) "Uma proposta metodológica para organizar o desenvolvimento de jogos originais", VideoJogos 2009
16. Joselli, Mark, Clua, Esteban (2009) "Mobile Game Development: A Survey on the Technology and Platforms for Mobile Game Development", VideoJogos 2009
17. Rising, Jim (2009) "Sashimi Waterfall Software Development Proccess", Managed Mayhem, http://www.managedmayhem.com/2009/05/06/sashimi-waterfall-software-development-process/ . (last accessed on June 14th, 2009)
18. Jacob, João,Coelho, António (2010) "Geo Wars – the development of a location-based game", Prisma 2010
19. João Jacob (2010) "[APP] Geo Wars - A location based game for the HD2" http://forum.xda-developers.com/showthread.php?t=694276 (last accessed on January 22th, 2011)
20. Rising, Jim (2009) "Sashimi Waterfall Software Development Proccess", Managed Mayhem, http://www.managedmayhem.com/2009/05/06/sashimi-waterfall-software-development-process/ . (last accessed on June 14th, 2009)
21. Borges, Ricardo, Augusto, Cícero, Volkweis, Mauricio, Konzen Andrea (2002) "Jogos em Inteligência Artificial", Universidade Luterana Do Brasil