Pervasive Games as Web-applications: A Case Study based on a Laser Game

Pervasive Games as Web-applications: a Case Study based on a Laser Game

Giovanni Delnevo
Dept. of Computer Science and Engineering
University of Bologna
giovanni.delnevo2@unibo.it

Luca Passeri
Two Year Master in Computer Science and Engineering
University of Bologna
luca.passeri3@unibo.it

Diego Pergolini
Two Year Master in Computer Science and Engineering
University of Bologna
diego.pergolini@unibo.it

Silvia Mirri
Dept. of Computer Science and Engineering
University of Bologna
silvia.mirri@unibo.it

***Abstract*—With the amazing success of PokemonGo, pervasive games have definitely caught the attention of the entire world. Their game experience is characterized by the fusion of real and virtual elements. In order to be able to extend the virtual world into the real one, such games have to access the mobile devices sensors like GPS and camera. For this reason, they are implemented as native or hybrid mobile applications. Contrary to this trend, in this paper we present Shoot Them All, a pervasive game implemented as a web application. In Shoot Them All, players have to hit the opponents, trying, at the same time, not to get hit. The battlefield is the real world, while the guns are virtual, as an extension of players' mobile devices. We were able to successfully developed such a game thanks to the availability of Javascript APIs that allow to access and manage device sensors. The success of this case study highlight how these technologies are mature enough, opening the door to web pervasive games.**

***Index Terms*—web pervasive game, web location-based game, web laser game**

I. Introduction

PokemonGo has become one of the most successful game ever [1], after being released in July 2016, making the pervasive games mainstream. One week after its publication, it counted 28.5 million daily unique players only in the United States. Thus, it has reached over 750 million downloads worldwide within its first year. The PokemonGo phenomenon has also attracted not only the interest of players, but also the one of the scientific community [2]. Many studies evaluated the consequences derived from its use [3]. Just to cite few examples, Zach and Tussyadiah evaluated its impact on the sense of community, mobility, and well-being [4], while Leblanc and Chaput suggested it as a successful strategy to increase physical activity levels of the populations [5].

Despite the recent exploit, pervasive games have been subject of studies for a long time, receiving continuously increasing attention, until becoming a popular field of investigation [6]–[8]. Different types of games have been grouped under this concept, as can be seen from a first overview presented by Magerkurth et al. [9]. They identified the following sub-genres of pervasive games: smart toys, affective gaming, augmented tabletop games, location-aware games, and augmented reality games [10]. This has caused a sort of ambiguity, which has resulted in the lack of a standard and shared definition of the term *pervasive game* [11]. However, generally, they can be defined as *games that combine together elements of the real, social and virtual world*. Valente et al. defined four boundary criteria to identify pervasive mobile games [12]. They have to: i) be context-aware, ii) use mobile devices, iii) access remote data, and iv) be multi-player. According to their vision, only the first two criteria are mandatory.

Both in the scientific literature and in the commercial world, we can find several examples of pervasive games. Among these, in addition to Pokemon Go, we can mention Pirates! [13], Pervasive Clue [14], Botfighters [15], Can you see me now? [16], Geocaching [17], Urbanopoly [18], GeoZombie [19], Ingress[1], up to the very recent Harry Potter: Wizards Unite[2]. The older ones, like Pirates! and Pervasive Clue, were implemented for ad-hoc devices, since they were developed in the pre-smartphone era. The newer ones, instead, are developed as native or hybrid applications. The reason is quite obvious and lies in the need to access the smartphones sensors [20], [21]. In fact, in order to mix the real world with the virtual one, application must be able to access to the position sensor, in the case of location-based games, or to the camera, for games employing augmented reality.

Nevertheless, in the last few years, W3C has released several specifications, with the aim of defining standard interfaces for obtaining information from smartphones sensors. Actually, there are specifications for the following sensors: accelerometer[3], ambient light[4], gyroscope[5], magnometer[6], orientation[7],

[1]https://www.ingress.com/
[2]https://www.harrypotterwizardsunite.com
[3]https://www.w3.org/TR/accelerometer/
[4]https://www.w3.org/TR/ambient-light/
[5]https://www.w3.org/TR/gyroscope/
[6]https://www.w3.org/TR/magnetometer/
[7]https://www.w3.org/TR/orientation-sensor/

and proximity[8]. Even if they are released as Working Drafts, they are increasingly being supported by many browsers. Obviously, from great power comes great responsibility. The access to smartphones sensors poses problems regarding security and privacy, as highlighted in [22], [23].

Such policy, undertaken by the W3C, follows the current trends of web access, which have seen, since 2016, the mobile and tablet Internet usage exceeds desktop computers[9]. This represents an interesting opportunity in the development of pervasive games. In fact, they can start to be developed as web applications rather than as native or hybrid mobile ones.

In this paper, we present *Shoot Them All*, a pervasive game, designed and developed as a web application, consisting in a mobile laser game. The idea is to create a virtual laser game immersed in the physical environment. The aim of the game is to hit as many opponents as possible, while avoiding, at the same time, being hit. The circular battlefield will be specified using latitude and longitude, representing the center, together with the radius, that determines its amplitude. Players will be allowed to move only within such area, looking for other players to shoot at. Players' smartphones will have a dual task. First of all, they will allow users to complete all the management activities like create new matches, join already existing ones, manage their personal profiles, and see the leaderboard of the game. The second and more important function will be act as a gun during the matches.

With regard to the four boundary criteria, cited above, that define pervasive mobile games, Shoot Them All respects all four of them. In fact, the physical position of the players will determine if players will be hit by bullets or not (first criterion), players will have to employ a mobile device (i.e., smartphone or tablet) to be able to play the game (second criterion), all the data relative to the game will be stored in a remote database, only accessible trough predefined API (third criterion), and the game will require at least two players (fourth criterion).

The challenge, here, is twofold. On the one side, we want to implement a game, that usually requires additional devices (such as laser tag), only with a smartphone. On the other side, instead, we want to develop a pervasive game, in particular location-based, as a web application, hence employing only standard web technologies. Although these ones present some limitations in the access to the device hardware, a web application presents different advantages. For example, it does not require any installation, ensuring, at the same time, the maximum compatibility with different devices.

The remainder of this paper is organized as follows. Section II describes the system architecture and the communication between clients, the server and the database. Section III is devoted to illustrate the implementation of the whole system, with a particular focus on the client, the server and the collision detection system. Section IV details the implemented prototype, showing different screenshots of the implemented web application. Then, Section V presents some limitations of the current prototype and some possible extensions. Finally, Section VI concludes the paper.

[8]https://www.w3.org/TR/proximity/

[9]http://gs.statcounter.com/press/mobile-and-tablet-internet-usage-exceeds-desktop-for-first-time-worldwide

II. System Architecture Design

This Section describes the outline architecture of the system, also reported in Figure 1. As shown, it is composed by three main entities: clients, a server and a database.

A client is any device equipped with sensors for positioning (i.e., GPS, accelerometer, gyroscope, and magnetometer). This is the minimum requirement for clients to play the games, since they are needed to compute collisions between the bullets of a player and his opponents. The server has two main tasks. The former one is to continuously update the database with the information about the matches and the players. The latter one, instead, is to provide all the connected clients with such updated information.

There are two different types of interaction between these two entities: (i) client-server, and server-client. With the first type of interaction, managed through a standard REST API, a client is able to: register into the system, login into the system, create a new match, join an existing match, and visualize the personal profile of a given user.

The second type of interaction, instead, is used to provide players with some real-time information. In particular, the following data have to continuously be sent to all the connected clients: the position of each player during a game, the active matches available in the system, the score of each player register to the platform, and the score of each player during a game.

Finally, the database simply contains the state of all the matches, the global score of each player and the score of the players during a match.

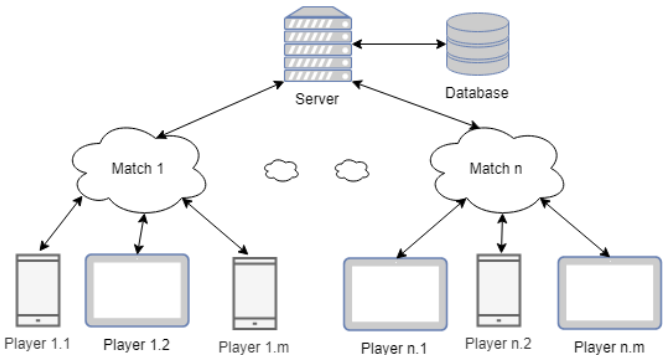

Fig. 1. System Architecture: Overview

III. Implementation

In this Section, we describe the implementation of the proposed system. We employed the MEAN stack [24] (Mongodb, Express, Angular, and Node.js), since it offers a modern and effective solution to realize web applications. The implementation of the server, of the client, and of the collision detection are discussed in isolation in the following Subsections.

A. Server

The server takes advantage of Node.js as a web server. In particular, the framework Express is used since it wraps many low-level Node functionalities, in a simplified and convenient way.

With regard to the client-server communication, described in the previous section, we implemented REST APIs that will be used by the client through Ajax calls. In particular, the implemented APIs expose five main routes:

- `/registration`: it allows users to register for the game, providing all the necessary data.
- `/login`: it lets the clients to authenticate themselves, providing username and password.
- `/profile`: it allows to get all the public information relative to a specific player.
- `/matches`: it allows to get the list of all the current matches or to create new ones. Moreover, for each existing match, different sub-routes are available. They allow to get/update, the state of the match, to get the list of participants, to add/remove players, and to get/modify a player's position and score.
- `/users`: it allows to get the list of the registered users and, for each of them, to get/update the relative score.

The server-client communication is instead implemented by exploiting web sockets, that ensure full-duplex real-time communications. In particular, we employed the Socket.IO library[10]. To facilitate the communication between socket.io process and non-socket.io ones, we took advantage of Redis[11], an in-memory data structure store, that can be used as a database, cache and message broker. The exchanged messages can be grouped in different topics:

- `users-pos`: this channel contains all the information about the position of each player during a match.
- `matches`: in this channel there are all the information about all the matches that are going on.
- `users-leaderboard`: in this channel all the information about the global scores of all the players registered are exchanged in order to update the global leaderboard real-time.
- `users-score`: the scores of all the players in a given match are sent in this channel to update the leaderboard of the match real-time.
- `time-out`: in this channel are sent all the messages that communicate a change in the match status (in preparation, started, ended).

Finally, to host the MongoDB database, we decided to take advantage of the Mlab service[12], a Database-as-a-Service platform. All the data are saved in this space, with the server that makes sure to update them every time they change, then notifying the affected clients.

Three main schemas were identified to map all the domain of the game. In particular, they are:

- `Users`: schema in which all the documents containing the information relative to the registered users are saved.
- `Users in Match`: in this schema there are all the documents concerning the participation of users in a match, such as the score, the position, and the belonging team.
- `Rooms`: here there are the historical data about all the matches created, like the position of the match, the maximum number of participants, and the match typology (i.e., public or private).

B. Client

The client side of the web application has been implemented using Angular [25], version 6. Following Angular guidelines, we used Angular Components to manage the presentation logic, while we added all the application logic and the interaction with the server in Angular Services.

The structure of the implemented Angular Components can be easily represented through a hierarchy of components, as shown in Figure 2. Each box in the picture represents a component. Some of them (i.e., Home and Match) have also child components. The arrows, instead, represent the navigation flow.

The main page, as expected, is managed by the *Home component*. Such a component presents a brief description of application (*Description*) and allows to visualize the global leaderboard (*Leaderboard*) and the available matches, presented both as a simple list (*Matches List*) and as a map (*Matches Map*). For the authentication, we implemented the *Login* and *Registration components*, that show, respectively, the login and the registration form. The *User Profile* component, instead, shows the user profile. A user can create a new match through the form of the *Match Configuration* component. Once a new match has been created or selected from the list, it is possible to visualize a summary of the match information (*Match Info*) and to join, if possible, that match. Finally, in the *Match component*, the actual game takes place (*Game Map*). During the match, it is possible to see also the scores of all the players that have joined the match (*Match Leaderboard*).

During the development of the components, we also took advantage of the following libraries in order to implement a User Interface modern and mobile-first: Angular Material[13], SCSS[14], ChartJs[15], iDangero.us Swiper[16], and Angular Google Maps[17].

C. Collision Detection

The collision detection system is based on two main data: the position and the orientation of each player in the match. These data simply come from the device sensors, in particular

[10]https://socket.io/
[11]https://redis.io/
[12]https://mlab.com/
[13]https://material.angular.io/
[14]https://sass-lang.com/
[15]https://www.chartjs.org/
[16]https://idangero.us/swiper/
[17]https://angular-maps.com/

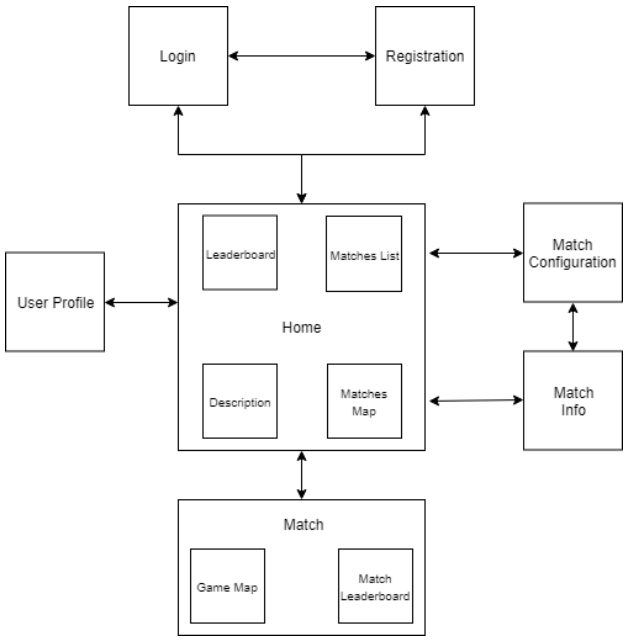

Fig. 2. Angular Components

GPS for the position, and accelerometer, gyroscope, and magnetometer for the orientation.

To obtain the position of a player, we used the HTML 5 Geolocation API. Such an API provide an object *Navigator.geolocator* with a method *getCurrentPosition* that returns an object containing both the current latitude and longitude of the device, together with the accuracy of position.

Instead, to get the orientation of the device with respect to the three Earth axis (x, y, and z), we simply registered a listener to the event *deviceorientationabsolute*. Every time that there are changes in the orientation, an object is returned. Such an object contains four fields: (i) *absolute*, that determines if the position is absolute or not; (ii) *alpha*, that is the rotation angle with respect to the Earth z-axis (where 0 is the East, 90 is the North, 180 is the West, and 270 is the South), (iii) *beta*, that indicates the inclination with respect to the Earth y-axis; (iv) *gamma*, angle of rotation with respect to the Earth x-axis. To determine the device orientation, we used only the alpha and beta information. However, the current prototype uses the beta information only to compute the orientation of the device, but it does not consider such information in the collision detection system (i.e., if a player A shoots to a player B that is right in front of him, but on a different floor, he/she will be hit). This means that, by now, the z-axis is not taken into account.

For performance reasons, we decided not to use the server to compute the collisions during the matches. It only takes care of the creation and management of the matches. Instead, all the game logic, including the collision calculation, is distributed to the clients (i.e., the players in the match). This way, the workload of the server is limited, improving the overall scalability of the system [26]–[28].

IV. Our Prototype

In this Section, we illustrate the main features of the current prototype. The relative code is currently maintained in a public Github repository[18].

Once logged in, a user has access to the homepage of the web application. It is composed by four different views, reported in Figure 3. Firstly, there is the global leaderboard of the system (case a), in which players are ranked, based on their scores. As shown, each player has also a *Ranking* title (e.g., Corporal and Master Sergeant), that is always computed based on the player score. In the second view, instead, a brief description of the game is reported (case b). Finally, the last two views provide the list of the matches, already created and that are still open, in two different ways. In the first one, the match are geo-localized on a map (case c), while in the other one the matches are represented as a list (case d). In both cases, a click on a match, will allow to visualize the detail of the match and, in case, to join it.

Besides joining an existing match, it is also possible to create a new one. In Figure 4, case a, we depict part of the creation process. During it, a player has to specify the match name, if the match will be public (everyone will be able to join it) or private (a password will be required in order to join it), some players settings like the team names, how long the match will last, and the match area. The match area will always consist in a circle, therefore requiring the coordinates (latitude and longitude) of the center and the relative radius (expressed in meters). Figure 4 also reports how the information of a match, once a player has joined it, are shown. Case b shows the waiting room before a match starts. In fact, each match, after the creation, will start in sixty seconds, giving to other players the time to join it. Once the match starts, players can choose the teams, if they were specified in the creation process (case c). In the leaderboard section, it is possible to visualize the team scores (case d), together with the player's score of each team.

Finally, in Figure 5, we report the main screenshots of the application during the game. It consists in: a radar, that shows other players nearby, the current player's score, the gun, that allows to shoot, and the remaining bullets.

V. Limitations and Future Works

Even if the current prototype allows to play an entire game against other players, it still presents two major limitations. The first one regards the collision detection system, as already highlighted in Subsection *Collision Detection*. In fact, the altitude of the players is not considered when computing collisions. The second one, instead, consists in the positioning system. Actually, the prototype exploits GPS to trace the player's position but it only works well in outdoor environments. The implementation of an indoor-positioning system would allow to play both in indoor and outdoor environments. However, a reliable indoor-positioning system is still an open problem. In fact, the several approaches presented in literature

[18]https://github.com/lucapasseri/Shoot-Them-All

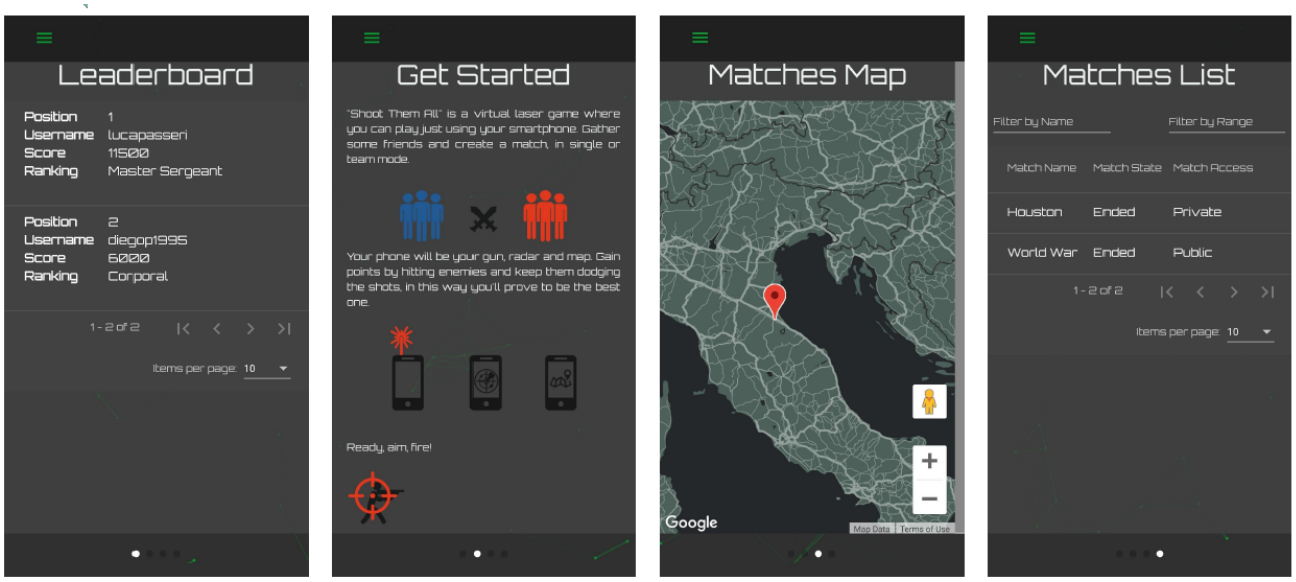

Fig. 3. Homepage screenshots: Leaderboard (a), Description (b), Matches Map (c) and Matches List (d)

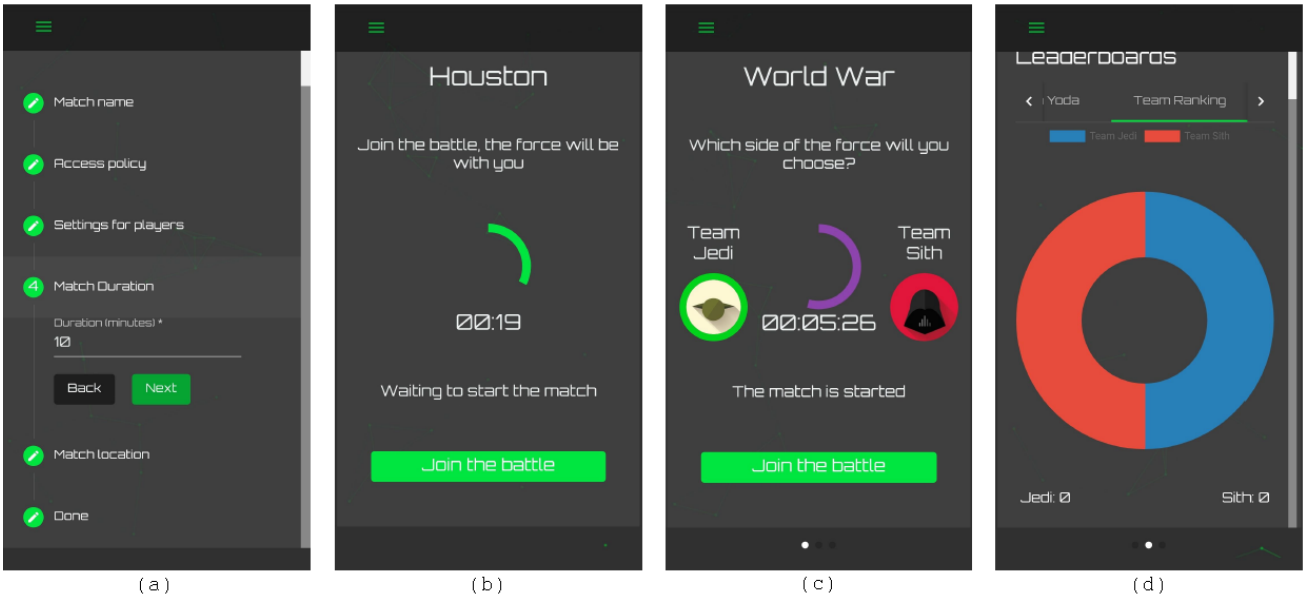

Fig. 4. Match Creation and Info

(e.g., Wi-Fi and Bluetooth based positioning, and magnetic field fingerprinting) still have an error of several meters [29].

Besides the limitations described above, we would like to implement and investigate the following additional features and improvements:

- Make the gaming experience more engaging. This could be achieved taking advantage of the human-centered design. An evaluation session with different users should be conducted. Then, the collected feedback should be used to improve the overall user experience of the system.
- Introduce more weapons. Currently, there is only one weapon, with a predefined range. Different weapons (e.g., pistols, assault rifle, tactical rifle, sniper rifle, ...) could be introduced, each of them with different range, available bullets, and recharge time.
- Providing communication mechanisms among players. As an example, a chat in the waiting room would allow a better organization of players in teams. Thus, during the match, a team chat would let players coordinate themselves. However, it would also require an adequate reporting system, that allows players to report the so-called *toxic* players [30].

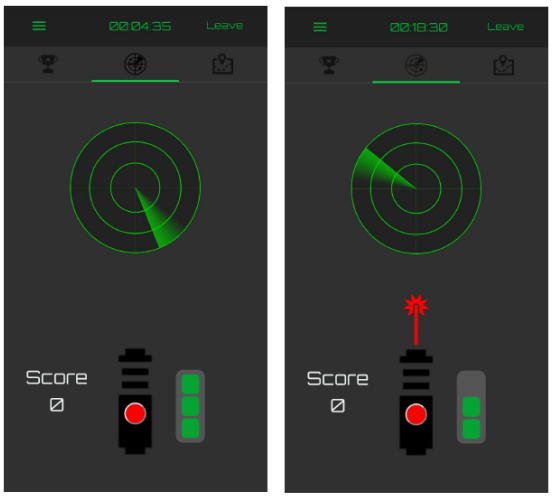

Fig. 5. Shooting System during matches

VI. Conclusion

Pervasive games have now become mainstream. Due to their functional requisites, they are often implemented as native or hybrid applications. In this paper, instead, we propose Shoot Them All, a pervasive game implemented as a web application. It consists in a virtual laser game, immersed in the physical environment, where people have to hit the opponent players without being hit. The successful development has been possible thanks to the growing presence of Javascript APIs that allow to manage devices sensors. Such a support opens the door to a new generation of pervasive games that can be now implemented as web applications.

References

[1] K. Alha, E. Koskinen, J. Paavilainen, and J. Hamari, "Why do people play location-based augmented reality games: A study on pokémon go," *Computers in Human Behavior*, vol. 93, pp. 114–122, 2019.

[2] A. M. Clark and M. T. Clark, "Pokémon go and research: Qualitative, mixed methods research, and the supercomplexity of interventions," 2016.

[3] V. R. Wagner-Greene, A. J. Wotring, T. Castor, J. K. MSHE, and S. Mortemore, "Pokémon go: Healthy or harmful?," *American journal of public health*, vol. 107, no. 1, p. 35, 2017.

[4] F. J. Zach and I. P. Tussyadiah, "To catch them allthe (un) intended consequences of pokémon go on mobility, consumption, and wellbeing," in *Information and communication technologies in tourism 2017*, pp. 217–227, Springer, 2017.

[5] A. G. LeBlanc and J.-P. Chaput, "Pokémon go: A game changer for the physical inactivity crisis?," *Preventive medicine*, vol. 101, pp. 235–237, 2017.

[6] L. Valente, B. Feijó, J. C. S. D. P. Leite, and E. Clua, "A method to assess pervasive qualities in mobile games," *Personal and Ubiquitous Computing*, vol. 22, no. 4, pp. 647–670, 2018.

[7] A. Bujari, M. Ciman, O. Gaggi, and C. E. Palazzi, "Using gamification to discover cultural heritage locations from geo-tagged photos," *Personal and Ubiquitous Computing*, vol. 21, no. 2, pp. 235–252, 2017.

[8] C. Prandi, V. Nisi, P. Salomoni, and N. J. Nunes, "From gamification to pervasive game in mapping urban accessibility," in *Proceedings of the 11th Biannual Conference on Italian SIGCHI Chapter*, pp. 126–129, ACM, 2015.

[9] C. Magerkurth, A. D. Cheok, R. L. Mandryk, and T. Nilsen, "Pervasive games: bringing computer entertainment back to the real world," *Computers in Entertainment (CIE)*, vol. 3, no. 3, pp. 4–4, 2005.

[10] P. Salomoni, C. Prandi, M. Roccetti, L. Casanova, L. Marchetti, and G. Marfia, "Diegetic user interfaces for virtual environments with hmds: a user experience study with oculus rift," *Journal on Multimodal User Interfaces*, vol. 11, no. 2, pp. 173–184, 2017.

[11] E. Nieuwdorp, "The pervasive discourse: an analysis," *Computers in Entertainment (CIE)*, vol. 5, no. 2, p. 13, 2007.

[12] L. Valente, B. Feijó, and J. C. S. do Prado Leite, "Mapping quality requirements for pervasive mobile games," *Requirements Engineering*, vol. 22, no. 1, pp. 137–165, 2017.

[13] S. Björk, J. Falk, R. Hansson, and P. Ljungstrand, "Pirates! using the physical world as a game board.," in *Interact*, vol. 1, pp. 423–430, 2001.

[14] J. Schneider and G. Kortuem, "How to host a pervasive game-supporting face-to-face interactions in live-action roleplaying," in *Position paper at the Designing Ubiquitous Computing Games Workshop at UbiComp*, pp. 1–6, 2001.

[15] O. Sotamaa, "All the world's a botfighter stage: Notes on location-based multi-user gaming.," in *CGDC Conf.*, Citeseer, 2002.

[16] S. Benford, A. Crabtree, M. Flintham, A. Drozd, R. Anastasi, M. Paxton, N. Tandavanitj, M. Adams, and J. Row-Farr, "Can you see me now?," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 13, no. 1, pp. 100–133, 2006.

[17] K. O'Hara, "Understanding geocaching practices and motivations," in *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 1177–1186, ACM, 2008.

[18] I. Celino, D. Cerizza, S. Contessa, M. Corubolo, D. DellAglio, E. Della Valle, and S. Fumeo, "Urbanopoly–a social and location-based game with a purpose to crowdsource your urban data," in *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pp. 910–913, IEEE, 2012.

[19] C. Prandi, P. Salomoni, M. Roccetti, V. Nisi, and N. J. Nunes, "Walking with geo-zombie: A pervasive game to engage people in urban crowd-sourcing," in *2016 International Conference on Computing, Networking and Communications (ICNC)*, pp. 1–5, IEEE, 2016.

[20] A. Melis, S. Mirri, C. Prandi, M. Prandini, P. Salomoni, and F. Callegati, "Crowdsensing for smart mobility through a service-oriented architecture," in *2016 IEEE International Smart Cities Conference (ISC2)*, pp. 1–2, IEEE, 2016.

[21] S. Ferretti, S. Mirri, C. Prandi, and P. Salomoni, "Trustworthiness in crowd-sensed and sourced georeferenced data," in *Proceedings of the 2nd International Workshop on Crowd Assisted Sensing, Pervasive Systems and Communications (CASPer 2015)-in conjunction with IEEE PerCom*, 2015.

[22] A. Das, G. Acar, N. Borisov, and A. Pradeep, "The web's sixth sense: A study of scripts accessing smartphone sensors," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1515–1532, ACM, 2018.

[23] M. Roccetti, S. Ferretti, C. E. Palazzi, P. Salomoni, and M. Furini, "Riding the web evolution: from egoism to altruism," in *2008 5th IEEE Consumer Communications and Networking Conference*, pp. 1123–1127, IEEE, 2008.

[24] S. Holmes, *Getting MEAN with Mongo, Express, Angular, and Node*. Manning Publications Co., 2015.

[25] Y. Fain and A. Moiseev, *Angular 2 Development with TypeScript*. Manning Publications Co., 2016.

[26] V. Ghini, P. Salomoni, and G. Pau, "Always-best-served music distribution for nomadic users over heterogeneous networks," *IEEE Communications Magazine*, vol. 43, no. 5, pp. 69–74, 2005.

[27] A. Bujari, M. Massaro, and C. E. Palazzi, "Vegas over access point: Making room for thin client game systems in a wireless home," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 12, pp. 2002–2012, 2015.

[28] S. Mirri, C. Prandi, P. Salomoni, F. Callegati, A. Melis, and M. Prandini, "A service-oriented approach to crowdsensing for accessible smart mobility scenarios," *Mobile Information Systems*, vol. 2016, 2016.

[29] P. Davidson and R. Piché, "A survey of selected indoor positioning methods for smartphones," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 1347–1370, 2016.

[30] J. Blackburn and H. Kwak, "Stfu noob!: predicting crowdsourced decisions on toxic behavior in online games," in *Proceedings of the 23rd international conference on World wide web*, pp. 877–888, ACM, 2014.