SoundPark: Towards Highly Collaborative Game Support in a Ubiquitous Computing Architecture

Romain Pellerin[1], Nicolas Bouillot[2], Tatiana Pietkiewicz[1], Mike Wozniewski[2], Zack Settel[3], Eric Gressier-Soudan[1], and Jeremy R. Cooperstock[2]

[1] Conservatoire National des Arts et Métiers, Paris, France
{romain.pellerin,eric.gressier_soudan}@cnam.fr

[2] Centre for Intelligent Machines, McGill University, Montreal, QC, Canada
{nicolas,mikewoz,jer}@cim.mcgill.ca

[3] University of Montreal, Montreal, QC, Canada
zs@sympatico.ca

Abstract. Ubiquitous computing architectures enable interaction and collaboration in multi-user applications. We explore the challenges of integrating the disparate services required in such architectures and describe how we have met these challenges in the context of a real-world application that operates on heterogeneous hardware and run-time environments. As a compelling example, we consider the role of ubiquitous computing to support the needs of a distributed multi-user game, including mobility, mutual awareness, and geo-localization. The game presented here, "SoundPark", is played in a mixed-reality environment, in which the physical space is augmented with computer-generated audio and graphical content, and the players communicate frequently over a low-latency audio channel. Our experience designing and developing the game motivates significant discussion related to issues of general relevance to ubiquitous game architectures, including integration of heterogeneous components, monitoring, remote control and scalability.

1 Introduction

Our research is oriented toward the use of distributed architectures to support high-level group interaction in computer-mediated environments, in particular, involving virtual and mixed reality. The applications we are exploring require an engaging level of interaction between multiple users and their environment, in which cooperation, coordination, and mutual awareness are critical. As we explain in further detail below, a ubiquitous computing architecture is not only

appropriate for such requirements, but offers several further benefits of mobility and context-awareness, where context includes location, time, user activity, physical environment [1], as well as execution environment [2].

Ubiquitous computing and physical interaction with mobile device have proven to be significant factors in improving social relationships in teams working toward a common goal. Examples include children playing [3] or tourist groups visiting an unfamiliar museum [4]. However, when team members interact remotely, mutual awareness is often highly constrained. Existing ubiquitous games have addressed this shortcoming in part, but with an emphasis primarily on location awareness. We believe that significant opportunities remain untapped to improve the communication between players and interaction with the environment. Specifically, these aspects of game play can benefit from the use of personalized rendering of audio and graphics displays that enable continuous awareness between the team members and of their augmented environment.

To support our exploration of this potential in the context of a new mixed-reality multi-player game, *SoundPark*,[4] we developed three novel services intended to improve the degree of engagement and interaction between players: a framework for immersive spatial audio performance; a low-latency, high-fidelity streaming engine; and a service-oriented middleware for ubiquitous games. SoundPark takes full advantage of these generic services, including structured session management, continuous audio and graphic rendering, sensor management and low-latency communication. As anticipated, the integration of these components resulted in an unavoidable increase in implementation complexity.

The remainder of this paper is organized as follows. Section 2 summarizes and compares the present work with other games in this category. Section 3 describes the details of our own service architecture, addressing the issue of complexity. Section 4 is dedicated to the design of SoundPark, and reviews some of the lessons learned from the associated implementation effort. Section 5 summarizes lessons learned and discusses the implications of this work to supporting continuous mutual awareness in a ubiquitous computing environment. Section 6 concludes with suggestions for the development of future ubiquitous game architectures.

2 Related work

The infrastructure of multiplayer ubiquitous games is typically based on computationally limited devices such as mobile phones, PDAs, and portable game systems, that delegate the complexity of group communication and state management to servers. In this context, mobile devices have traditionally enabled ubiquitous mixed reality by providing geo-localization, either to provide players' position or to collect virtual objects in the environment [5]. Examples include Botfighter [6], CanYouSeeMeNow [7], Mogi [8], and CatchBob [9].

A seminal example is Cheok's *Human Pacman* [10], which inherits its design from the eponymous arcade game of the early 1980's, but consists of human

[4] A sample video is available at `http://www.audioscape.org/twiki/bin/view/Audioscape/MobileAudioscape`.

players wearing head-mounted displays (HMD) and playing the role of *pacman* or *ghosts*. Human Pacman has been extended by different projects, including Pac-Lan RFID [11], where both game objects (pills, super-pills, and ghosts) and human players have associated RFID tags. Such tags, increasingly popular in entertainment applications, are used for location, identification and content delivery [12, 13]. In addition to simple geo-localization, certain recently proposed games, such as MeetYourHeartTwin [14], propose the additional integration of biosensors to establish social relationships between players. In this example, players are able to use their PDA displays to see others who share their heartbeat characteristics, e.g., within a similar frequency range.

Table 1 provides a general description of previously mentioned games. While various localization technologies provide more or less accurate tracking, team coordination is typically restricted to simple mechanisms such as position visualization, text messages or standard walkie-talkies.

The architecture presented here shares many characteristics with these earlier works. However, rather than focusing only on visualization of position, we emphasize audio interaction for communication between players, for rendering various aspects of game state, and for enabling decision-making related to choices that affect the virtual world. Further, in contrast with the games described above, where little contextual information is exchanged between players, we integrate localization, low latency audio and other sensorially acquired data to support richer mutual awareness. In turn, our experience highlights the important tradeoff between wireless communication over a large physical scale, where network access may be discontinuous, and tasks mandating a high level of connectivity. Needless to say, this remains a challenge that future related middleware will continue to face.

At the level of interaction design, our approach is distinguished by its emphasis on continuous modeling and perception of the overlaid virtual world. The positions and states of virtual objects are dynamically updated and may be manipulated through lightweight, mobile devices. Moreover, the nature of continuous audio perception, as it relates to geo-localization of the players, requires significantly greater accuracy of sensing than provided by GSM network cell identification, RFID or standard GPS, in addition to nearly continuous network connectivity.

3 Architecture

Support for mutual awareness and context awareness as relevant to user-level collaboration requires integration of various functionalities, leading to a complex combination of hardware and software components. This necessitates an extensible architecture, so that missing functionality can be enabled as required, and raises several design decisions that must be addressed. Foremost among these is the tradeoff between a distributed computing ideal, with all computation performed in a distributed manner, versus a more centralized, and likely less scalable, architecture. While high-end mobile devices may have the neces-

sary processing capacity to support general-purpose applications, most highly portable platforms remain less suitable for the demands of real-time computation, such as user-specific multi-source audio spatialization or merging of virtual graphical content with the physical world in a mixed reality framework.[5]

Faced with the processing constraints of mobile devices, we were required to proceed with a centralized architecture, allocating the demanding computations to more powerful servers. On the positive side, our service-oriented approach enables extensions and supports reusability of already implemented components. In particular, our architecture allows for the separation of application- and device-specific code. As can be understood from the following description, this facilitates rapid prototyping of many applications requiring mutual awareness between users and context-awareness of both their states and that of the environment. Moreover, because the middleware provides connectors to handle data from any type of sensor, it is well suited for the integration of mobile devices with additional sensor inputs.

3.1 Overview

Our architecture is shown in Figure 1. The server is dedicated to global state management, user-customized audio, and graphical rendering of the virtual environment. It consists of two components: Audioscape for real-time rendering of audio and 3D graphics, and the uGASP server, based on our middleware that targets ubiquitous multiplayer gaming. The uGASP server, compliant with the OSGi specification[6], is the master component of the architecture. It handles high-level application events, that can be triggered by Audioscape for rendering the virtual environment. Existing uGASP services are used for coordinating scenario, session, and sensor management with a game engine.

Mobile clients support sensing of the user and the surrounding environment, as well as local delivery of audio and graphics. User input components, for physical mobile interaction, include GPS, Wiimote and RFID support. Although presented together in Figure 1, due to the limited functionality of current mobile products, the client features are actually deployed on two separate mobile devices, as described in Section 4.2.

3.2 uGASP

The uGASP[7] [15] middleware implements the Open Mobile Alliance Games Services (OMA GS) working group specifications that deal with multiplayer

[5] The Human Pacman game [10] (see Section 2) is no exception: although local 3D visual rendering is performed locally to the user, the equipment includes a personal computer carried in a back-packed and a HMD. These requirements dramatically reduce the user's long-term mobility.

[6] OSGi Service Platform Release 4 specifications, `http://www.osgi.org`

[7] uGASP is available under the L-GPL license from `http://gasp.objectweb.org/ubiquitous-osgi-middleware.html`

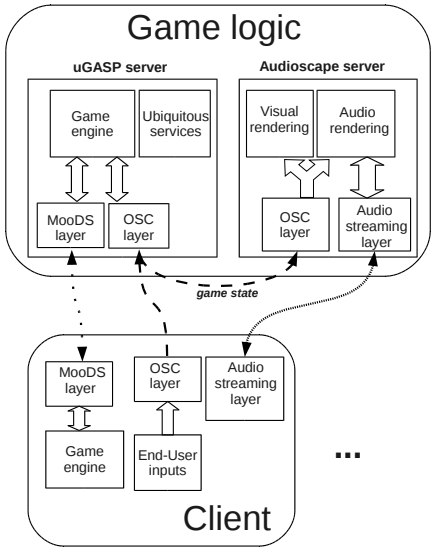

Fig. 1. SoundPark Middleware Architecture Overview.

game management on embedded devices. uGASP, implemented in Java J2ME, is based on the Open Services Gateway Initiative (OSGi) component framework, and more specifically on the iPOJO OSGi layer that provides additional modular, dynamic and configurable services through the creation, deployment and calling of a bundle[8]. Interestingly, this allows for deployment, optionally dynamic, of specific optimized instances of the middleware, freeing the binary code from unwanted functionality.

uGASP is composed of multiple families of services, including network communication, session management, a game server engine that handles game logic, ubiquitous services and system services. Game logic is implemented as a Java class that implements the Server interface. During implementation of the server, the required uGASP services are specified by the developer. These are then instantiated automatically as appropriate during initialization of a game session.

Existing bundles have been extensively employed in server and client implementations. For example, the uGASP location manager bundle is used to compute projections of player positions for both 2D and 3D displays, described later. Another bundle was deployed on mobile phones for handling RFID tags. The middleware was also improved with a bidirectional network communication connector, supporting more efficient exchanges than were available with the pre-existing HTTP connector. As seen in Figure 4, this new connector provides

[8] In OSGi terminology, a software component providing a specified service is denoted as a bundle.

the communication layer used between the mobile phone and the Gumstix computer. Similarly, an OSC protocol (see Section 3.4) bundle was developed for the purpose of handling data acquired from various sensors and interpreting events necessary to manage the scenario.

uGASP, in addition to being a key part of the SoundPark architecture, provides a bundle for deploying a client side game engine. A 2D map display on mobile phones extracted from our SoundPark prototype illustrated by Figure 2, is used to show the locations of other players and virtual objects. In this application, the deployed byte-code contains only the necessary bundles, providing an optimized application memory footprint, as required for deployment on the resource constrained device.

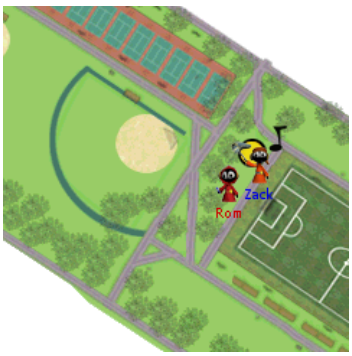

Fig. 2. A 2D virtual world view appearing on a mobile phone display, extracted from our SoundPark prototype. Shown are two player avatars, a sound loop virtual object indicated by the musical note, and a player (Zack), approximately in the middle of the goal line of the football field (anchored at his feet). The yellow circle represents the staging area.

3.3 Audioscape

Ubiquitous mixed reality interaction leads naturally to context aware rendering of graphical and audio content. This is delegated to the *Audioscape* platform [16] that models the location of predefined sound sources, including both virtual objects and users, in a 3D representation of the physical environment. Using game events received through the uGASP server, the Audioscape server's role is to maintain the state of the virtual 3D environment, and compute spatialized audio renderings for each user, including real-time processing based on directivity of sound specification, radiation pattern, and realistic effects such as reverberation and Doppler shift.

Despite its name, Audioscape is also used for 3D *visual* rendering of the scene for monitoring the application, and optionally for visualization by an audience, as described below. This is accomplished with a customized graphics engine built with OpenSceneGraph[9], which represents users and virtual objects as arbitrary 3D models, as illustrated in Figure 3. This allows for monitoring of the actions of users, represented as dynamic 3D avatars, and the application status, on a virtual display.

[9] `http://www.openscenegraph.org`

Fig. 3. 3D virtual world view, in which two players, the hunter and scout, can be seen. Sounds are spread over the field.

In their current version, the Audioscape 3D audio and graphical engines are integrated into Pure Data (PD) [17], a pseudo real-time environment dedicated to (simulatenously) programming and executing interactive multimedia applications. The PD internal design provides an interface allowing the development of additional processing components, called *externals*. At the client side in our architecture, our PD externals are executed by *Pure Data anywhere* (PDa) [18], a rewritten fixed-point version of PD.

3.4 Communication

Communication between software running on the servers and mobile devices uses three different protocols. The first one, the MooDS protocol [19], providing binary serialization process within uGASP, is dedicated to optimized and reliable object-oriented communication with J2ME enabled devices. It was therefore considered suitable for game event dissemination.

Open Sound Control [20] (OSC) is a character-oriented data exchange protocol for real-time control of media processing, (initially sound processing) that has gained considerable popularity in the distributed audio community. It provide namespaces and is optionally encapsuled inside the UDP protocol, making it suitable for rapid prototyping of applications that require low-latency transmission of data, but that can tolerate sporadic data losses. This is particularly well-adapted for periodic data, such as those acquired by sensors.

Finally, the high-fidelity audio streaming protocol, *nStream*,[10] provides bidirectional low-latency audio communication. Its development was motivated in a large part by the high latencies and computational complexity associated with typical compressed audio formats, and the unsuitability of traditional RF

[10] *nStream* is a Pure Data component, compatible with Pure Data's fixed-point version. *nStream* source code, distributed under the GNU General Public Licence, is available within the Audioscape SVN repository, accessible at `http://www.audioscape.org`.

solutions given their limited range, high power draw, and signal interference issues. Low-latency,[11] generally required for remote audio interaction, is particularly important in our architecture, where all user audio rendering and team communications are computed remotely by the Audioscape server.

4 SoundPark

In order to evaluate our architecture, when supporting a deep level of player collaboration in a ubiquitous game context, we enabled very precise location tracking, powerful yet power frugal mobile computing resources, and low-latency audio connectivity between the players. This section describes the SoundPark game design and how it emphasizes player collaboration in a mixed reality environment, then covers the technical game implementation.

4.1 Game design

The SoundPark game was designed for the Jeanne Mance park, an area covering 630x190 meters in the city of Montreal. The physical park environment was augmented with (physical) RFID tags and (virtual) audio objects, which were initially placed at pre-determined locations. Consistent with the theme of a mixed reality environment, game objects typically exist as both real (physical) and virtual entities: clues are represented by their associated bollards and RFID tags, but reveal their content graphically on a mobile phone display. Sound loops are only heard by users via directed listening, i.e. when close to the corresponding location, and only once the associated clue has been discovered. Although sound loops are only perceived in the virtual domain through the players' headsets, they can also be moved from place to place. In addition, audio spatialization is used to guide users toward the sound loop location, which becomes better audible with the use of volume level mapped with user's actual distance to the virtual sound location.

A team is composed of two player types, whose combined goal is to create a musical arrangement of multiple sound loops, assembled in a staging area, or home base. Walkie-talkie-like communication is used to coordinate their activities. Action is initiated by the *scout*, who discovers clues by reading the RFID tags distributed throughout the game area, and then observes the associated sound loop locations through a mobile display that also provides continuous updates of player locations (see Figure 2). Once the clue has been discovered, the associated sound loop appears in the virtual world, and can be heard by *hunters* who pass near to its physical location. The scout's role is then to guide the hunter near the sound loop, which is optionally acquired and then attached to the hunter, who must carry it to the staging area. After the sound loop is deposited, it becomes part of the target musical arrangement, composed of the

[11] With nStream, and by using recording software to capture both the original audio source signal as it is supplied to the mobile sender and the output audio from the receiver, we measured a total end-to-end delay of 14.4ms.

combination of all sounds acquired by the team. The game ends when one of the teams has successfully collected all its required sound loops.

To increase the challenge of the game, clues indicate the location of individual sound loops, but not their musical properties, which can only be discovered by the hunters as they draw near. Furthermore, not all sound loops will fit with the theme of a target musical arrangement. For example, a baroque guitar loop cannot be part of a jazz arrangement. Attempts to deposit such a sound loop in the staging area fail with an appropriate feedback message.

4.2 Implementation

During development, we extensively employed the services and components of our architecture. The game design itself was programmed with only a few (455) lines of code within the server, taking advantage of game engine and management services provided by uGASP. The most demanding part of creating the game itself was modeling and calibrating the park for 3D and 2D visualization, followed by creating the audio content.

In terms of user interface hardware for the players, our game requires only small form factor devices, such as the Gumstix, described later, and near-field communications (NFC)-enabled Nokia 6131 mobile phones. Players in SoundPark use role-specific hardware: a *scout* player carries a mobile phone that displays a map of a section of the game environment, and with an NFC-RFID reader. Although RFID tags could contain game information in their own memory,[12] we simply use their Universal Identifiers (UID) to index data into the game database, hosted on a server. On the other hand, *hunters* do not require mobile phones, but rather, play with a Wiimote controller used for sound loop acquisition and deposit, accomplished by pressing the buttons on the controller.

As seen in Figure 4, we selected the Gumstix computer platform[13] as the central mobile device for all players. It consists of a Verdex XM4-bt main board with expansion boards for memory, 802.11g (WiFi) communication, bidirectional high-quality audio I/O, and differential GPS. It supports Linux and is powered by a portable USB power pack. Connectivity to other devices (e.g., mobile phone and Wii controller) has been enabled via Bluetooth.

Our software hosted by the Gumstix was implemented in C, as a set of PD externals (see section3.3). We used our already existing, adapted and cross-compiled externals including the OSC protocol, Wiimote controller access, and nStream engine. The code for handling GPS's standard NMEA data format has been added for the purpose of game development.

5 Lessons learned and discussion

The SoundPark implementation has effectively shown that our architecture requires few lines of code to develop a complete, working application. In addition,

[12] We are using Mifare 1k RFID tags, which provide 1 kB of storage capacity.

[13] `http://www.gumstix.com`

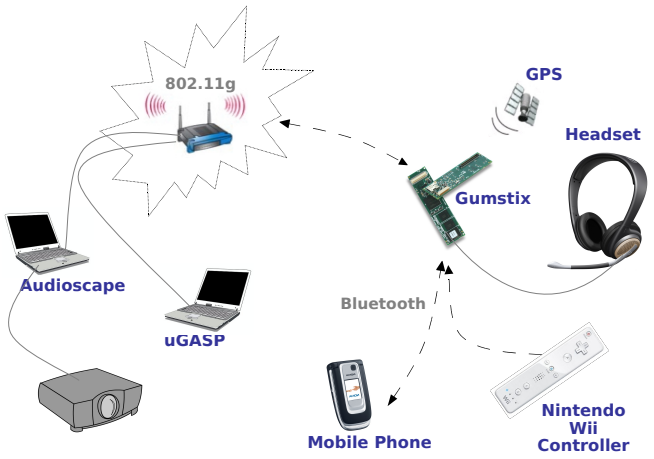

Fig. 4. SoundPark devices overview. Note that several of the components are unique to particular player roles (e.g., mobile phone for scouts, and Wii controller for hunters).

services added to the architecture are reusable for future application implementation. While this significantly reduced the necessary effort during application design and deployment, many issues remain. In this section, we discuss various related aspects of our experience: the use of mobile devices for the user interface, difficulties encountered during physical world modeling, integration, control and monitoring of the software components, and scalability.

Mobile devices for the user interface Mobility requirements imposed hard limits regarding weight, size, and power autonomy. Solutions had to be found to extend battery life for all of the the various components, ranging from the Gumstix to the WiFi network and field servers, while still maintaining adequate processing and transmission power. In terms of logistics, the equipment *worn* by players should be mounted solidly and be well-protected from physical damage, but still have sufficient ventilation so as to not overheat. Our prototyping experience quickly demonstrated that sealed boxes of electronics and a hot summer day do not mix well. Ideally, the gear should be fixed securely to each player, yet still be easy to put on and take off. Future experiments will need to better cope with the constraints introduced by wearable computing hardware [21].

Modeling physical world In order for 3D modeling to serve as an effective representation of the game state, each physical element (tennis courts, baseball diamond, walking paths, football posts, etc.) had to be modeled precisely with respect to its actual dimensions and physical location in the game area. Otherwise, the 2D and 3D displays, and in particular, the placement of player avatars within it, would not correctly correspond to the observations of the players and

audience when viewing the real, non-virtual park. This effect, in addition to having a potential negative effect on the spatialized audio rendering, could also impact communication within the team.

Once the physical world is modeled, players are virtually placed within the 3D model, with their positions continuously updated via conversion of actual GPS coordinates to the model coordinate system. However, the uneven nature of GPS performance must be taken into account. Although our differential GPS sensors provide reasonable precision for most areas of the game environment, the extracted positions from GIS maps (such as those provided by Google and Yahoo!) lead to mapping errors with our measurements. This, in turn, frustrates our manual process of calibrating between the 2D (mobile phone display), 3D (Audioscape representation), and GPS coordinates, as needed for a consistent rendering of the players and their environment.

Integration Due to the many disparate technologies involved, our middleware, uGASP, has to link together all these components and continuously compute the game state. However, several components, operating autonomously, are linked to uGASP via communication protocols, disabling their monitoring and control by uGASP internal interface. In contrast to uGASP components, nStream, Audioscape and PDa externals that access the GPS and the Wiimote devices are controlled and monitored through their own interface.

Global observation of all of the components is continuously enabled, in addition to uGASP built-in monitoring provided by the iPOJO layer, throughout the 3D environment. For instance, sensor states, such as player positions, are displayed, and 3D graphical vu meters are used to monitor audio communications. Our experience has shown that these metaphors are useful for understanding the status of the system. While our current implementation integrates mainly high level observations, such as user positions, it can be easily extended with lower layer ones, such as GPS signal conditions, and thus possible loss of accuracy. With such data attached to the client avatar and displayed on a dedicated 3D rendering, monitoring of the entire application can be achieved using traditional game-like navigation in the virtual environment.

The control and orchestration of the overall application were also significant challenges: control mechanisms for the various components include manual user control, OSC messages, and ssh. This heterogeneity complicates the task of orchestrating the full life cycle of an entire game. We are currently developing an orchestration component that abstracts access to, and functionality of, the components within the framework of a high level control infrastructure.

Scalability issues As mentioned in Section 3, the computational restrictions of most existing mobile devices leads to a centralized architecture, where user-personalized audio rendering and game management are achieved with a server. Evolving towards a more distributed architecture, where mobile devices will operate autonomously by rendering the game state locally will significantly enhance scalability, since the number of users will not be limited by server capabilities.

Although it would improve scalability, a more distributed architecture will also introduce additional challenges for global state management, player communication, consistency, and overall orchestration and coordination. These issues will become increasingly significant as more players participate in the game.

Moreover, in SoundPark, client-server interaction is supported by a single wireless network configured in managed mode that covers the entire game space. While robustness was sufficient to support the traffic generated by our application, this single network architecture restricts the game space to a static area. This constraint motivates us to investigate a more dynamic configuration that uses an ad-hoc networking strategy, as well as higher level protocols for managing group communication and low-latency transmission of data on overlay networks. In addition, and toward a more scalable and autonomic solution, we are also investigating adaptive protocols for the exchange of periodic data (audio and sensor), possibly at non-trivial bandwidth, and that better tolerate network performance variations and disconnections.

6 Conclusion

We presented a ubiquitous game architecture that eases the integration of heterogeneous components required for highly collaborative applications. This was used to support our mixed-reality game, SoundPark, which runs in a fully continuous and physically modeled environment with multimedia content and communication. The game further integrates user-personalized audio spatialization and low-latency audio streaming. As SoundPark delegates the players to different social roles, it vividly illustrates the possibilities of a ubiquitous computing architecture to support the demands of a highly engaging social activity.

SoundPark required only three months of development, taking advantage of the many reusable components of our service-oriented architecture. This amount of time is remarkably short, especially given that the members of the SoundPark team had not worked all together on any previous projects. Although we implemented several components of the architecture, many of them are reusable, while game-specific code contains few lines.

Finally, we provided details about lessons learned: in addition to several application specific issues, we discussed integration of heterogeneous components, their monitoring and control, as well as architecture scalability.

References

1. Schmidt, A., Beigl, M., Gellersen, H.W.: There is more to context than location. Computers and Graphics (1999) pp. 893–901
2. Schilit, B., Adams, N., Want, R.: Context-aware computing applications. Workshop on Mobile Computing Systems and Applications, 1994. (Dec 1994)
3. Leichtenstern, K., Andr, E., Vogt, T.: Role assignment via physical mobile interaction techniques in mobile multi-user applications for children. In Heidelberg, S.B.., ed.: Ambient Intelligence. Volume 4794/2007 of Lecture Notes in Computer Science. (2007) pp. 38–54

4. Stahl, C.: The roaring navigator: a group guide for the zoo with shared auditory landmark display. In: MobileHCI '07: Proceedings of the 9th international conference on Human computer interaction with mobile devices and services, New York, USA, ACM (2007) pp. 383–386
5. Natkin, S., Yan, C.: Adaptive narration in multiplayer ubiquitous games. IJCiNi (2007) IGI Publishing.
6. Sotamaa., O.: All the world's a botfighter stage: Notes on location-based multi-user gaming. In by Frans Mäyrä. Tampere, E., ed.: Proceedings of Computer Games and Digital Cultures Conference *CDGC'02*, Finland (June 2002)
7. Flintham, M., Benford, S., Anastasi, R., Hemmings, T., Crabtree, A., Greenhalgh, C., Tandavanitj, N., Adams, M., Row-Farr, J.: Where on-line meets on the streets: experiences with mobile mixed reality games. In: CHI '03, Proceedings of the SIGCHI conference on Human factors in computing systems, New York, USA, ACM (2003) pp. 569–576
8. Joffe, B.: Mogi, Location and presence in a pervasive community game. In: Proceedings of the Seventh International Conference on Ubiquitous Computing, Ubicomp. Ubiquitous Computing, Entertainment, and Games Worshop, Tokyo (September 2005)
9. Girardin, F., Nova, N.: Getting Real with Ubiquitous Computing: the Impact of Discrepancies on Collaboration. eMinds **1** (2006)
10. Cheok, A., al.: Human Pacman: a mobile, wide-area entertainment system based on physical, social, and ubiquitous computing. Personal and Ubiquitous Computing (May 2004) pp.71–81(11)
11. Rashid, O., Bamford, W., Coulton, P., Edwards, R., Scheible, J.: PAC-LAN: mixed-reality gaming with RFID-enabled mobile phones. Comput. Entertain. (2006)
12. Bohn, J.: The smart jigsaw puzzle assistant: Using RFID technology for building augmented real-world games. In: Workshop on Gaming Applications in Pervasive Computing Environments at Pervasive. (2004)
13. Heumer, G., Gommlich, F., Mller, A., Jung, B.: Via mineralia - a pervasive museum exploration game. In: 4th International Symposium on Pervasive Gaming Applications, PerGames. (2007) pp. 159–160
14. Flammer, I., Ka, W., Skraba., R.: Meet your heartbeat twin. In: Proceedings of 4th International Symposium on Pervasive Gaming Applications PerGames 2007, Salzburg, Austria (June 2007) page 157–158
15. Pellerin, R., Gressier-Soudan, E., Simatic., M.: uGASP: an OSGi based middleware enabling multiplayer ubiquitous gaming. In: International Conference on Pervasive Services, ICPS 2008 Demonstration Workshop, Sorento, Italy (July 2008)
16. Wozniewski, M., Settel, Z., Cooperstock, J.R.: A paradigm for physical interaction with sound in 3-D audio space. In: Proceedings of International Computer Music Conference (ICMC). (2006)
17. Puckette, M.: Pure Data. In: Proceedings of the International Computer Music Conference, San Francisco (1996) 269–272
18. Geiger, G.: PDa: Real time signal processing and sound generation on handheld devices. In: International Computer Music Conference (ICMC). (2003)
19. Pellerin, R.: The MooDS protocol: a J2ME object-oriented communication protocol. In: Mobility '07: Proceedings of the 4th international conference on mobile technology, applications, and systems, ACM (2007) 8–15
20. Wright, M.: Open sound control 1.0 specification. Published by the Center For New Music and Audio Technology (CNMAT), UC Berkeley (2002)
21. Piekarski, W., Thomas, B.: Arquake: the outdoor augmented reality gaming system. Commun. ACM **45**(1) (2002) 36–38

game	**description**	**localization technology**	**networking**
BotFighters [6]	multiplayer city game using text messages to seek enemies. Browser game interface is provided for audience in order to monitor the game state	GSM network cell identification	GSM network
CanYouSeeMe [7]	multiplayer outdoor game using 2D graphical display on a PDA to seek other players. Audio communication between player using walkie-talkies	GPS	Wireless LAN
CatchBob [9]	multiplayer game where users collaboratively discover information necessary to catch Bob, a virtual object. Information is localized and displayed on a tablet PC using a 2D map overlay	radio beacon	Wireless LAN
Myht (Meet your heartbeat twin) [14]	multiplayer game continuously using players' heart rates to determine couples that have to meet. PDAs are used to display twin location	GPS	General Packet Radio Service (GPRS)
Pac-Lan [11]	multiplayer campus game using RFID tags to seek other players. For localization, graphical 2D display on mobile phone is provided to players and audience	RFID-based	General Packet Radio Service (GPRS)
Human-Pacman [10]	augmented reality (3D overlay on real images) campus game using HMD and backpack computers. Human helpers communicate with player using text messages to help them find each other. Bluetooth cookies are used for real world augmentation.	GPS	Wireless LAN

Table 1. A comparison of notable mobile pervasive computing games.