A Software Platform for Developing Multi-player Pervasive Games Using Small Programmable Object Technologies *

Orestis Akribopoulos, Dimitrios Bousis, Dionysios Efstathiou, Haris Koutsouridis, Marios Logaras, Andreas Loukas, Alexandros Nafas, Georgios Oikonomou, Irini Thireou, Nikos Vassilakis, Panagiotis Kokkinos, Georgios Mylonas and Ioannis Chatzigiannakis

Computer Technology Institute (CTI) and CEID, University of Patras, 26500, Greece
E-mail: ichatz@cti.gr

Abstract: In this paper we present a platform for developing mobile, locative and collaborative distributed games comprised of small programmable object technologies (e.g., wireless sensor networks) and traditional networked processors. The platform is implemented using a combination of JAVA Standard and Mobile editions, targeting also mobile phones that have some kind of sensors installed. We briefly present the architecture of our platform and demonstrate its capabilities by reporting two pervasive multiplayer games. The key characteristic of these games is that players interact with each other and their surrounding environment by moving, running and gesturing as a means to perform game related actions, using small programmable object technologies.

1 Introduction

As of 2008, the total number of mobile phone subscribers has well surpassed the number of 3 billion. Along with the increase in the number of subscribers, there has been an increase of the capabilities of such devices. The vast majority of the current generation of mobile phones are capable of executing J2ME applications. Moreover, manufacturers have started integrating various kinds of sensors into their handsets, e.g., accelerometers or thermistors. Therefore, there is already an existing user base using such devices, that is continually growing. It is our belief that there is great potential in combining sensors and mobile devices to produce exciting entertainment applications. Games have been a major part of the computer industry for the last decades, and are generally recognized as a means of pushing the technological boundaries, both in hardware and software. We expect that pervasive games will transform into a major application field for wireless sensor networks.

So far, the potential of combining gaming and mobile devices with sensing capabilities has not been yet studied in great depth. Although there have been some attempts to develop multi-player games that rely on devices sensing the real world, these works are limited in number and scope, and very few software environments exist to aid developers in combining these two areas. General issues related to the development of J2ME games using bluetooth connectivity are discussed in [5]. Examples of games combining WSN with multi-player games are [3, 4, 2]. Such games range from virtual pet games, to cooperative hidden treasure games and to multiplayer serious tourism-focused games. There are also some examples of using 3-axial accelerometer sensors to control 3D games.

In this paper we present a software platform that aims to help developers in building multi-player pervasive games through the use of mobile devices and the combination of wireless sensor networks. Our design goals are: *generality*, providing a framework that allows a wide range of games to be developed; *ease of programming*, to reduce the development cycle of games; *scalability*, to allow large numbers of users to participate in the same game; *heterogeneity*, to support different technologies; *openness*, to be open and extensible to new technologies and game domains.

These goals reflect observations on the state and future trends of pervasive multiplayer games. We use our platform to implement two example games, which we call *Moving Monk* and *Assassin Apprentice*. We believe that these examples reveal the variety and the joyfulness of games that can be produced though our platform. The use of our platform is rather intuitive and does not distract the developer from his main task, that is, the design, development and testing of exciting game applications.

2 Overview of our Platform

The architecture of our platform is based on a hierarchy of layers where each layer is comprised of one or more peers. Each such layer is assigned a particular role in the

*This work has been partially supported by the IST Programme of the European Union under contract number IST-2005-15964 (AEOLUS) and by the ICT Programme of the European Union under contract number ICT-2008-215270 (FRONTS).

1-4244-2575-4/08/$20.00 ©2008 IEEE

game. Each peer may be a traditional networked processor or a small programmable object technology. In the following paragraphs, we briefly describe the layers of our platform (see Figure 1).

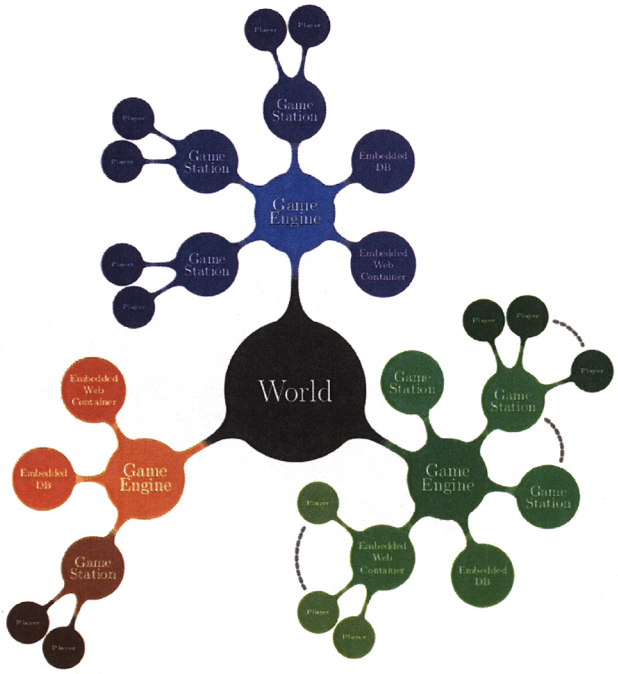

Figure 1. Overview of the architecture

The *World* is a central peer that is accessible via the Web and offers high level services. It maintains the player records, gathers player related statistics and keeps track of game's history. Players can interact with the game system, view statistics related to previous and in progress games, access web services for interaction with external social networks. The database maintained by the World peer is updated asynchronously, whenever it is possible without causing computational overhead to the game engines.

The *Game Engines* are peers that coordinate single game instances. Each game instance is assigned to a specific engine. They are the master peers that control all interactions and keep up-to-date information related to the specific game via an embedded database. The engine processes all events generated by the game (i.e., enforce game rules) and store them to the embedded database, which is always synchronized with the data stored on the players' devices. Periodically it also communicates with the World peer to update the global database. Engines provide game-specific services for implementing game scenarios and offer web services specific to a game for players controlling mobile devices. They are accessible either directly from the Internet (e.g., via UMTS) or via local ad-hoc networks (e.g., via 802.11).

The *Game Stations* are peers that control a specific physical area and connect to a Game Engine. Each engine may support multiple stations depending on the physical area covered by the game. These peers implement location-aware and context-aware services. The stations communicate regularly with the users' devices either through local ad-hoc networks or via personal area non-IP networks, providing rich gaming experience.

The *Players* are peers that are executed within the devices controlled by the users of a particular game instance. They communicate directly with the game stations and depending on the available networking capabilities they can also communicate with the engine. They are used to monitor the evolution of the game and interact with nearby players. When a player is found they may prompt the user for further actions. The user can use the sensors and buttons to explicitly trigger actions for interaction. Each action will cause the exchange of data and may involve neighboring players. Player peers provide services that allow them to interact even when they are disconnected from the network infrastructure for extended periods of time.

The design decision of introducing multiple layers may add some amount of complexity to the system. The gain however is that it introduces some crucial advantages: it allows the implementation of games that require near real-time response to specific aspects of the game (e.g., role playing, adventure and sport games); it offers location oriented services for games that need players to visit specific places (e.g., puzzle, strategy and treasure-hunt games); it enables the development of services that can augment the physical reality (e.g., via Google Earth); it allows games to be played by devices that do not necessarily offer Internet connectivity.

3 Implementation Overview

Various different technologies of wireless sensor devices exist, targeting low energy consumption, sensing modularity and reliable communication. Recently, Sun Microsystems developed the SPOT platform [1] that attempts to overcome the challenges that currently inhibit development of tiny sensing devices. The device is built upon the IEEE 802.15.4 standard. It is a small, battery operated, device running the Squawk Java Virtual Machine (VM) without an underlying OS. The novelty is that this VM acts as both an OS and software application platform, thus allowing programming of the devices in the Java Micro Edition platform. In addition, the developer can take advantage of all standard Java development tools in order to create applications. This platform has the potential to dramatically affect the nature and type of wireless sensor network applications.

Given this new hardware platform, we implemented our platform using the Java and the available technologies. Java

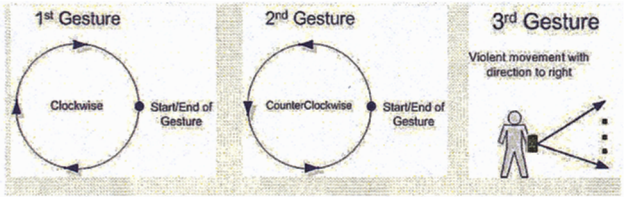

Figure 2. Sample gestures for player actions

is a powerful programming language with a large code base and a number of very efficient technologies (e.g, RMI, Server Pages, Persistence, J2ME) that we utilized. Java is used in all the layers of our architecture, providing homogeneity and easiness in the maintenance and extension of the code. SUN SPOTs, on the other hand, provide a flexible hardware and software platform for developers to experiment, prototype whatever they can imagine and innovate. It includes a very simple interface (two buttons and a number of LEDs), a variety of sensors (e.g., accelerometer, thermistor) and they are programmed using the J2ME. We believe that it would be quite difficult to develop our platform using mobile phones, mainly due to the lack of homogeneity and their cost [5].

4 Available Games

We have used our platform to implement the following two multi-player pervasive games. Both games implement services that allow the player peers to localize in indoor environments, perform tasks while on the move, coordinate their actions and allow delay tolerant communication.

Moving Monk: The idea of the game is that players (*the monks*) have to find the locations of the stations (which we call *the temples*) and to perform specific actions (which we call *prayers*) inside them. A Temple is defined by the coverage range of a game station. Temples can be visited in any order and specific gestures must be used, representing actions related to the game. Available gestures are depicted in Figure 2. The first monk to visit all temples and pray is the winner of the game. Players can access the web interface of the Engine peer in order to monitor their progress. Other users can be notified about the progress of the game by connecting to RSS feeds. Also users can view the physical position of the players via the well-known Google Earth application. The World peer provides KML feeds for visualizing the position of the active players.

Assassin Apprentice: The idea of this game is that one player is the *master* and the others are the *apprentices*. The players are not aware who is their *master*. The goal of the *master* is to hide her role and locate the apprentices and eliminate them by performing specific gestures. The goal of the *apprentices* is to prolong their participation in the game by avoiding the *master*. If the *apprentices* uncover the *master* they can eliminate her by combining their powers, that is, perform specific gestures simultaneously. Players will be notified of recent *eliminations* via the Engine's web interface (RSS feeds). External observers can observe the progress of the game via the Google Earth application.

We believe that these two games reveal the variety and the joyfulness of the games that can be produced using our platform. The *Moving Monk* game features location-aware services. The *Assassin Apprentice* game demonstrates player interaction, offers context-aware services and supports delay tolerant networking.

Currently, we are in the process of organizing our platform in a easy-to-deploy package. This will also include extensive documentation about the architecture and use of our platform. Moreover, we will resolve license-related issues, so as to make our system publicly available to the community. Finally, we will implement other games that we believe are even more exciting and fun to play.

Acknowledgments

This platform was partly implemented by undergraduate students during the Spring Semester 2008 in the Department of Computer Engineering and Informatics, University of Patras, Greece, as a part of the Distributed Systems II undergraduate class. We wish to thank Athanasios Gerostamoulos, Ioannis Patlakas, Dimitrios Petrochilos, Aourela Sehou, Panagiotis Tsirigotis and Lampros Tsolakos, that also participated in the class, for useful ideas.

References

[1] Sun™SPOT). http://www.sunspotworld.com/.

[2] R. Ballagas, A. Kuntze, and S. P. Walz. Gaming tourism: Lessons from evaluating rexplorer, a pervasive game for tourists. In *Proc. of 6th Intl. Conference on Pervasive Computing*, pages 244–261. Springer-Verlag, 2008. Lecture Notes in Computer Science, LNCS 5013.

[3] L. Liu and H. Ma. Wireless sensor network based mobile pet game. In *NetGames '06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, page 30, New York, NY, USA, 2006. ACM.

[4] S. Mount, E. Gaura, R. M. Newman, A. R. Beresford, S. R. Dolan, and M. Allen. Trove: a physical game running on an ad-hoc wireless sensor network. In *sOc-EUSAI '05: Proceedings of the 2005 joint conference on Smart objects and ambient intelligence*, pages 235–239, 2005.

[5] A. I. Wang, M. S. Norum, and C.-H. W. Lund. Issues related to development of wireless peer-to-peer games in j2me. In *AICT-ICIW '06: Proceedings of the Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services*, page 115, Washington, DC, USA, 2006. IEEE Computer Society.