RAPID PROTOTYPING OF LOCATION-BASED GAMES WITH THE MULTI-USER PUBLISHING ENVIRONMENT APPLICATION PLATFORM

R. Suomela[1], K. Koskinen[2] and K. Heikkinen[2]

[1] Nokia Research Center, Finland
[2] Lappeenranta, University of Technology, Finland

ABSTRACT

Location-Based Applications (LBA) react and adapt to changes in the environment of a user. Building LBAs, however, is a time consuming task, and a lot of effort needs to be put to the infrastructure supporting the development process. In this paper, a framework for rapid prototyping of location-based games (and applications) with the Multi-User Publishing Environment (MUPE) application platform is presented. The application platform allows developers to quickly prototype game ideas, and they can concentrate on the content of the games and game logic, rather than technology. The platform is further studied with a 24-hour development session, in which the developers were able to produce many functioning game applications in the given time frame.

1. INTRODUCTION

Mobile devices readily available affect how people spend their time. The mobile phones allow the users to make phone calls, connect to the Internet, and play games in a transparent manner. The mobile phone is always with the user, which opens up several possibilities for improving the use of the device and applications.

The device and applications can react to context, i.e. they know the user's situation and can adapt their behavior based on this. Special examples of this case are the Location-Based Applications (LBA). LBAs are applications that use location as an input to adapt their content and behavior. LBA are envisioned to be a major area of new application development, as most mobile phones can be positioned, and an enormous device base already exists.

Developing a location-based application, however, is hard work. A lot of effort has to be put to building the infrastructure, as well as its testing. Developing from scratch is a very time consuming task and there is a definite need for a tool for rapid prototyping location-based applications, in order to evaluate the application concepts before putting a lot of effort into them.
This paper applies the Multi-User Publishing Environment (MUPE) Suomela et al (1) application platform for rapid prototyping of location-based games and applications. In this work, MUPE is extended with a server side context producer; to create a framework for generating location based games and applications. MUPE implements virtual world that can be customized for each service. As a single context producer can be used in many applications, this framework allows reuse of existing software without changes, if the same positioning technology is used. If a new positioning method is used, a new context producer needs to be created. The framework was tested in a 24-hour code camp, in which several location based games were developed.

The paper is organized as follows. We start by looking at related work in the area. After this, MUPE platform, context information, and our extension for the location-awareness are introduced. Next, the coding session is presented, and finally the paper is concluded.

1.1 Related Work

Context-Awareness refers to the applications ability to react to a varying use situation. Dey (2), has presented a good definition of context: *Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the applications themselves.* In other words, any information that varies can be used to help the applications.

A very great challenge for any context-aware application is the versatility of the contextual information. This versatility can be further applied by context-aware application or services in a personalized manner. According to ePerSpace (3) the context can be roughly classified into following categories:

- Environmental context (e.g. light, humidity)
- Personal context (e.g. pulse, retinal pattern, mood, stress)
- Task context (e.g. explicit tasks, events)
- Social context (e.g. buddy-list)
- Spatio-temporal context (e.g. location, time, speed)
- Device context (i.e. device capabilities)
- Service context (i.e. service specific information, e.g. name)
- Access context (i.e. networking characteristics, e.g. traffic performance)

Context-aware content adaptation (to personal preferences, e.g. location) is addressed in many research communities. For example, the ePerSpace project (3) tries to tackle this issue. Hawick and James (4) introduce a concept that integrates user preferences to the contextual information. In this concept the mobile application changes its behavior based on both user preferences and contextual information. Content adaptation from the context-awareness point of view has also been studied by Kolari et al. (5). In (5), a platform for context-aware services was designed and implemented. Another article by Tamminen et al. (6) shows how different aspects of mobile contexts are created and maintained by situated actions in everyday life.

According to Di Flora et al. (7), context-aware applications need a service infrastructure that is capable of providing uniform context abstractions. It also reveals that plenty of software architectures have been proposed to provide applications that use context information. It also presents a modular service infrastructure that can support mobile context-aware applications. In (7), the infrastructure is divided into two layers of internal context (context within the mobile device and application context) and external context sources (e.g. weather).

Another context related publication by Korpipää et al. (8) concentrates on building the higher-level context abstractions. It focuses on extracting relevant context information based on the mobile terminal capabilities. Their context management framework consists of the context manager, resource server, context recognition server and application. In their framework the context manager acts as the central server and other instances of the framework act as clients.

Location-Based Applications (LBA) are a special case of context-awareness; LBAs use location information in the application logic. Location information is represented with a reference system. The reference system can be for example coordinate based, e.g. the reference system used in GPS (Global Positioning System) where position information is presented with *latitude*, *longitude* and *altitude*. Cell based positioning systems might provide only a unique cell-id so the reference system is a one-dimensional list of cells. Location information becomes meaningful when applications add meanings to the location information. For example the area covered by a WLAN (Wireless Local Area Network) cell could have the meaning ``Railway station''. The railway station could also be presented as a bounded area in the coordinate system of the GPS. Determining the location of an object in the reference system is the job for a positioning system. The following presents a rough division of how an LBA can acquire location information from an object:

- External infrastructure determines the position of the located object. These systems can be subscription based from a service provider, such as cell-based positioning provided by a mobile phone operator.
- Located object detects its own position. For example, an object uses an attached GPS module for getting location information.

There are many variations from these two methods, and a good survey of the location systems can be found in Hightower and Borriello (9).
The applications can utilize the location information in many ways; see for example Suomela and Lehikoinen (10).
Context-Aware games utilize location or context information for mixing the gameplay with real world related information. Treasure hunting games; such as Geocaching (11) have been around for a long time. Pirates! (Björk et al 12), ARQuake (Thomas et al 13), BotFighters (It's Alive, 14), Mogi (Newt games, 15), and Can You See Me Now? (Blast Theory, 16) are more recent examples of new context-aware games. These games mix the real and virtual worlds in new ways providing the players with a unique and new approach to gameplay. For analysis on how to use the real world as the game arena, refer to Sotamaa (17).

2. DEVELOPING LOCATION BASED SERVICES WITH MUPE

Multi-User Publishing Environment (MUPE) is a client-server application platform for creating multi-user context-aware applications for mobile phones. The platform is implemented with Java technology. The mobile phone has a J2ME (Java 2 Platform, Micro Edition) client supporting J2ME Mobile Information Device Profile (MIDP) version 1 and 2. The server is written with J2SE (Java 2 Platform, Standard Edition). The MUPE architecture can be seen in Figure 1. All parts of MUPE are available under the Nokia Open Source License version 1.0a (18).

The MUPE server is the container for the applications and it is the only component requiring modifications in most applications. A MUPE server implements a virtual world, and there is one server for each application. New applications are created by extending the basic MUPE virtual world (which consists of objects such as room, user and item) to the needs of the new application; see for example Suomela et al (19). MUPE server is influenced by the Multi-User Dungeons (MUD) and MUD Object Oriented (MOO) (20), which are fully functional virtual worlds that can be customized with a custom programming language.

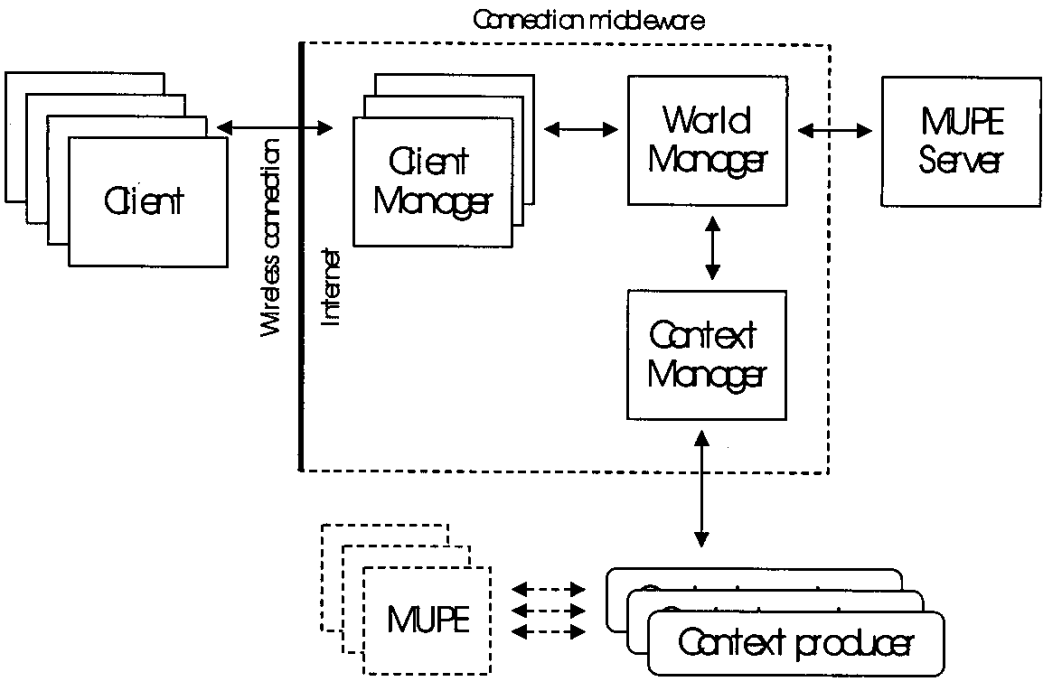

Figure 1: Overview of MUPE.

A single client in a mobile phone can be used to access all MUPE applications. The client creates custom UI and functionality with XML (eXtensible Markup Language) UI scripts. The script tries to wrap most of the MIDP functionality, thus allowing the developers to test mobile multi-user J2ME applications with scripts only. The scripts provide a fast way for prototyping new UIs, and reuse the existing UIs, as the client does not need software installs and code modifications. MUPE server separates the dynamic and static parts of the code, and the static XML UI scripts can be changed in the server and sent to the client without recompilation.
All the connections of MUPE components are handled by the platform connection middleware. At the moment, HTTP (Hypertext Transfer Protocol) and TCP (Transmission Control Protocol) connections are available for the mobile client, and new ones can be added as needed. All the connection types are available simultaneously, so the applications can decide what to use.

As a summary, MUPE has several features that enable rapid prototyping. Prototypes aid the development process, and validate concepts before starting the actual major development effort. Rapid prototyping, see for example Luqi and Steigerwald (21), aims at improving the analysis and design of specific systems. However, in this paper rapid prototyping is applied to the LBA-based game design and analysis within the MUPE platform. (21) Also points out, that software reuse is an excellent way for rapid prototyping. MUPE provides scripted UI development, ready-made connection framework, and code reuse in the form of a complete virtual world to speed up the development process. A more complete introduction to MUPE can be found in (1).

2.1 Context Producers in the MUPE framework

Each end-user client can feed location information to the MUPE system. The J2ME Location API (JSR-179) can be used in the client for location information, but at the time of the event, we had no devices with the J2ME API. In MUPE, context information does not need to originate from the mobile device, as any external information source can be connected as a context producer. The context producers provide an easy way to add server-side context information to the MUPE system. Context Producers act as external message brokers for the MUPE middleware: they convey context information from the originating systems (e.g. positioning systems, sensors...) to the MUPE applications. A single context producer can provide context information to many MUPE applications, as indicated in Figure 1.

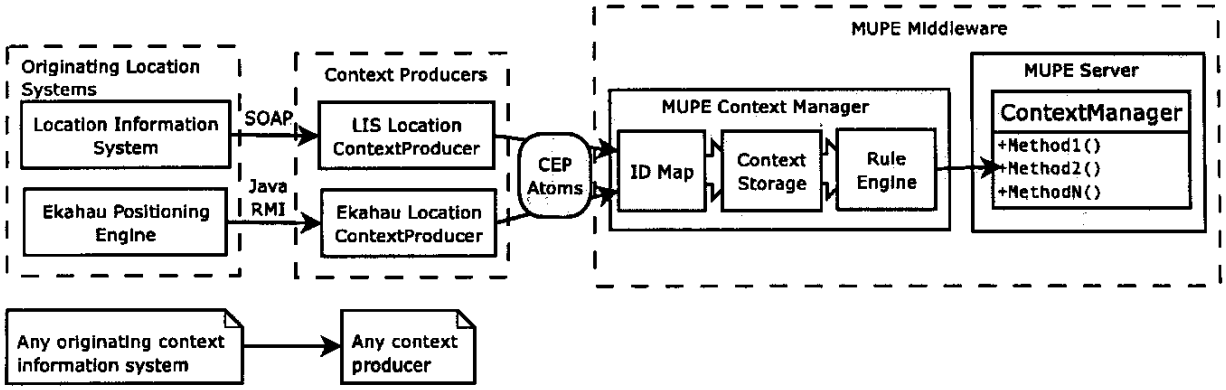

Figure 2: Context producer information production.

The context information is encoded with the Context Exchange Protocol (CEP), refer to Lakkala (22). Each service can then customize the use of the CEP information with Context Scripts; refer to Lakkala (23) and Java programming. CEP and Context Scripts allow each MUPE application to have a common way to encode and use context information, and thus reuse the context producers in every MUPE application. Context production related to the MUPE middleware can be seen in Figure 2. with gray color.
The process of communicating context information is as follows:

1. **Context Information** is created in the originating system, e.g. with sensors.

2. **Context Producer** creates a new CEP atom from the context information and forwards it to the MUPE context manager.

3. **Context Manager in middleware** does three things:
   a. The incoming context atom ID can be mapped to the corresponding ID in the MUPE server. As stated before (2), context information is related to an entity, so here we can map a service provider ID to an ID inside the virtual world. An example situation maps an atom related to a phone number to the user ID in the application.
   b. The data is stored in the Context Manager.
   c. The rules are checked to see if a method in the MUPE server needs to be called. Simple inference can be applied with the rules.

4. **ContextManager object in MUPE Server** executes the method call, and provides the information to the application logic.

This process allows different applications to reuse the same context producers, and customize the behavior with scripts only. For example, a context producer could be producing complex weather information. Different applications are interested in different parts of the atom; so one application could monitor temperature only, while the other would only look at several values and infer when a storm occurs. Each application could use the same context producer, but they would only use a part of the information. The applications would customize the Context Manager in the middleware to filter the necessary information and make appropriate method calls to the MUPE server.
The MUPE server would then be customized according to the application needs. The ContextManager in the MUPE server is a Java class, that has access to all of the world content, and it can act as an information storage component, or it can do more complex tasks as for example moving end users in the virtual world, or creating virtual world content.

2.2 Provisioning of Location Information (from a WLAN network)

MUPE application platform enables context-aware applications, but it does not contain any ready-made context-aware components. For the purpose of this paper, we implemented two context producers for acquiring context information from two originating systems (Ekahau and LIS) and thus enabling the implementation of location-based applications. The Ekahau Positioning Engine (EPE) (24) developed by Ekahau and Location Information System (LIS) developed in the WLPR.NET project (25). Figure 2. shows (as white area) the relation of the software components used in transferring and applying the location information from the WLAN positioning systems.

Both "Ekahau Location ContextProducer" and "LIS Location ContextProducer" served as protocol converters by communicating the location data from the respective positioning systems with SOAP (Simple Object Access Protocol) in LIS and Java RMI (Remote Method Invocation) in Ekahau into the CEP. The context producer components mapped the location data provided by both location systems and translated this data into the CEP format. LIS provided only cell location data, whereas Ekahau provided x/y coordinates, area locations and also the speed and heading the WLAN devices. The Context Producers acted also as multiplexers by taking the location data from both LIS and Ekahau and feeding it to multiple MUPE application servers.

Two types of scripts for the ContextManager were used. First, the identifier mapping scripts map the tracked device MAC (Media Access Control) addresses (identifiers used by the Ekahau positioning system) to the names given to the devices. These mapped names were used inside the applications to update the location of the correct user. Second, the Context Scripts execute a method call in the MUPE virtual world for transferring the location information for use in the application logic.

The ContextManager class in the MUPE server received this information, and updated the virtual world situation accordingly. In the beginning, the virtual world is empty. Every new user, who connects to the system, creates a new user object. Every time new location information arrives, the respective user object is located and updated (x/y coordinate, speed, heading, and area name). If the area name is new, a new room is created to the virtual world, and if a corresponding room already existed, it is looked up. After this, the user is moved to the corresponding room, if needed. This allows the same setup to be used at any location, as there is no predefined structure in the world, and it is filled as the users explore it.

As a summary, the MUPE server contained an object for each user device, and its x/y location. The user objects are moved between the server Rooms that correspond to the names of the real world rooms according to the context producer information.

3. DEVELOPMENT ASSEMBLY - CODE CAMP

Rapid prototyping and development of LBAs with the MUPE environment was tested in a 24h (17:00-17:00) intensive development session. This session, Code Camp, was organized in conjunction with the 2nd Workshop on Applications of Wireless Communications (WAWC'04) (26). Code Camp is aimed mainly at students interested in software development and has a variating topic each year. This year there were 18 participants; all of them (graduate or postgraduate) students apart from two who had a Ph.D. degree in Computer Science. The combining factor was that the participants didn't have prior experience with the MUPE platform.

At first, a 90-minute introduction to the event, the MUPE platform, an example game and a presentation of the WLAN positioning system arrangements were given to the participants. A goal to design and implement a game or an application (and have some fun) was set. Then the participants were freely divided into groups of 2-3 persons to begin the work. 90 minutes at the end of the 24-hour session was reserved for presenting the applications, and games. Also, the organizers selected the best game/application award.
As basis to build applications, the participants had the virtual world that created new areas as a user explored the real world. Also a team of people with prior experience with the MUPE platform was present in the event to help the participants during the development session. After the 24h session the best application was selected and rewarded.

Figure 3. shows the placement of WLAN access points in the Code Camp area floor plan. The emphasized areas present room locations, which were defined with the Ekahau Positioning Engine (24). The WLAN network build in the WLPR.NET project (25) was used to provide wireless and Ethernet network connectivity. Also two Linux workstations served as Bluetooth access points for Bluetooth phones.
The target client devices of the MUPE platform are J2ME mobile phones, and J2ME emulators were used in testing the applications. The emulators were installed in the workstations and the laptops. Development of the MUPE applications was first done in the workstations and then tested with the laptops. The laptops and PDAs were used as locatable objects in the MUPE applications. The location of the devices was derived from the WLAN network, which presented a practical problem, as most mobile phones do not have a WLAN radio. Also the MUPE client needs a MIDP environment and a suitable MIDP environment was not found for the PDA devices. So only the laptops served as real test target devices, because the MUPE client could be run with the emulator and location information could be acquired from the same device running the MUPE client. The PDA devices were only used as locatable objects. In addition the MUPE platform provides simulators to simulate the feeding of CEP data. Simulators were used on the workstations before making actual test with the client devices.

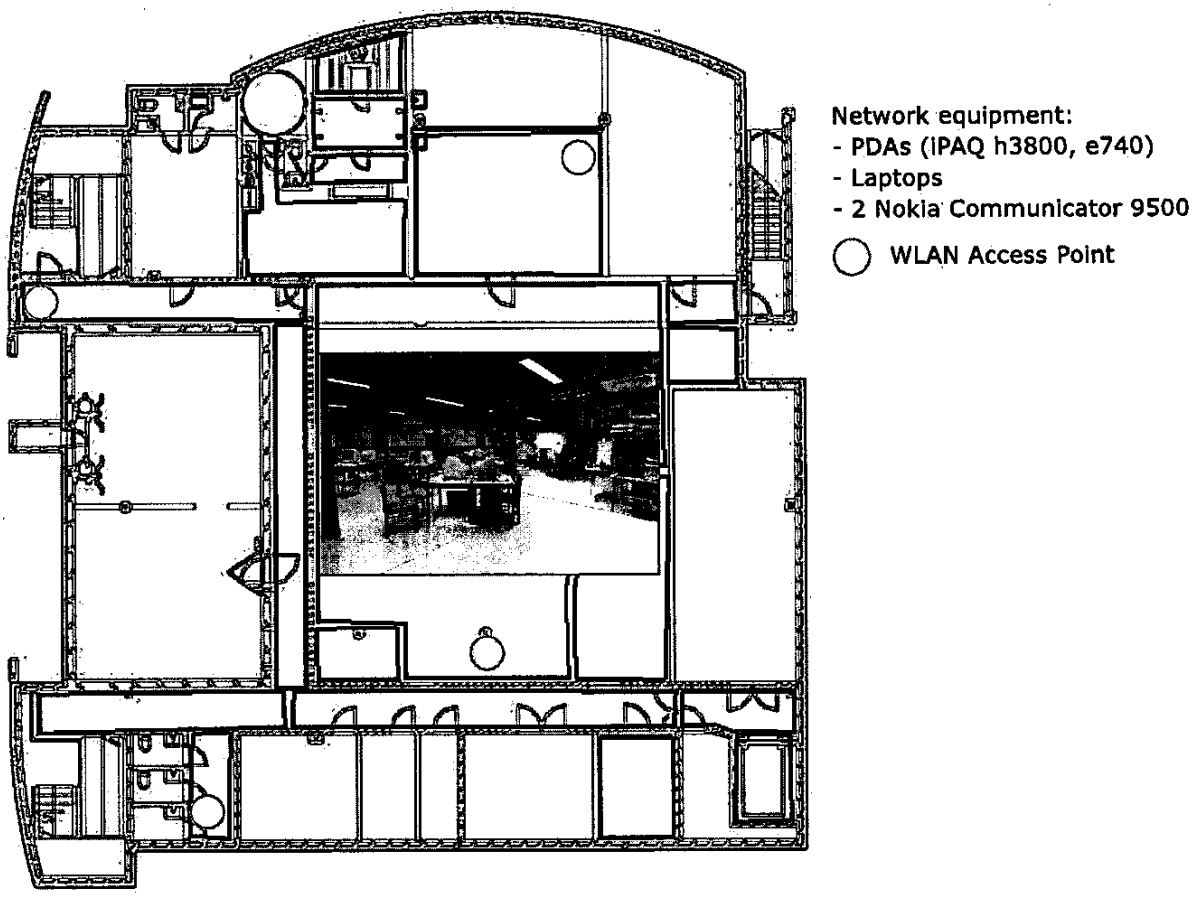

Figure 3: The code camp event area.

3.1 Positioning Arrangements

The WLAN network was used to provide location information of the WLAN client devices (PDAs, laptops). The Ekahau Positioning Engine (EPE) provides location information of the WLAN client devices with accuracy up to ~1m. The Location Information System (LIS) provides only cell based location information (4 unique cell locations in the Code Camp area). EPE has a two phase setup for acquiring location information (signal map construction and positioning based on the signal map).

Both positioning systems have some limitations concerning the accuracy of the location information. The LIS provides only rough cell based location information, but this information is available from any device equipped with a WLAN radio (providing that the device can associate with a WLAN access point). The LIS gathers sightings of the WLAN client devices from the WLAN access points as they roam from one WLAN access point to another. No additional software is needed in the client devices.

The Ekahau positioning system performs better in accuracy because it associates the RF signal environment to the actual physical environment. The client devices need to be able to constantly perform measurements of the WLAN RF signal environment. Monitoring software needs to be present in every client device. Also the signal environment should have enough variation (number of WLAN access points, observed signal strength) to reach a good steady accuracy with the EPE. This was a problem especially in the central area of the event, see Figure 3., because the signal environment did not have enough variation which led to an error greater then the mentioned ~1m.

3.2 Results from the Code Camp

The following paragraphs briefly describe the games developed in the Code Camp. All games were designed, implemented and presented in the 24-hour time limit. All the applications are available in the WAWC code Camp 2004 website (27).

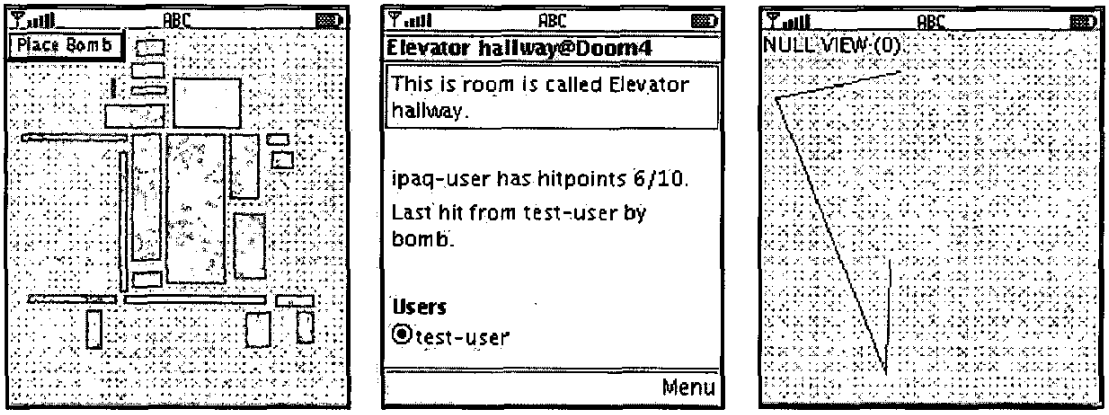

Figure 4: Bomber, Doom 4 and Nibbles -game demos (emulator screenshots).

Tag is a game that was implemented prior to the Code Camp as an example for the participants, and it was presented in the introduction of the camp. The game was very simple: if a player's Tag status is true, the player reverses all other players Tag status when entering a new room.

Bomber *-game centers on the classic idea of the Bomberman -game. The game draws a simplified map of the Code Camp area to the screen of the MUPEClient. New rooms are drawn when the users move into them (so they have to first explore their surroundings).*
The color of the room changes dark when a user moves into the corresponding room, see Figure 4. When two or more (up to four) players end up in a same room, the color of the room is drawn darker. Players can set a bomb in the rooms they visit. Points are gathered from blasting other players with bombs. A player scores when he/she manages to set a bomb into a room and blast the players in the room (and get away from the blast him/herself).

Doom 4 game is a shooter up game for MUPE, see Figure 4. This demo used both room and coordinate based location data. The players in the game have a collection of weapons at their disposal. Some weapons allow shooting and the player must aim at a correct direction to lower the hit points of the targeted player. Area weapons have a wider effect; a player can place bombs, mines or nukes in the rooms of the Code Camp area. Mines are triggered when a player steps on them, bombs can damage both the target and the player who places the bomb. Nuke is the solutions for a desperate player as it kills the player himself but damages all the players in the Code Camp area. However, a problem with the directional weapons is that the orientation of the client device is not easy to deduce as no map was presented to the user (nor is the client device capable to deduce its own heading).

Nibbles is a variation of the worm game found in most mobile phones, see Figure 4. Players who move in the Code Camp area leave a trail in their tracks. The goal of the game is to trap the opponents into a loop of their own tracks. A player should try to move without colliding with his or her own tracks.

Lovepiipper demo is an application for finding a suitable partner. When a player joins to the lovepiipper service, he or she specifies their sex, a description and what kind of things the player is looking for from a partner. The players can then move freely and when the service notices that a player with a matching profile is in the same room, a notification is displayed to both players.

DoctorTag is a variation of the tag game. A player affected with the tag can spread the tag to other players when moving from room to room. The game has a doctor who is able to cure the tag. Also the demo used the Big Screen Client of the MUPE to display the game events on a public screen. This allows people who are not in the game to follow the events in the game (in for example a cafe).

3.3 Evaluation of MUPE platform, and applications

The code camp allowed us to evaluate two things. First, what did the developers think about the platform, and how easy it was as a development platform. Second, the games and applications themselves can be evaluated.

3.4 Development

During the code camp, the organizers assisted the developers in their work. This served two purposes: First, to assist the programmers, and second, to allow the organizers to evaluate the MUPE platform in our task. During the development, we constantly discussed

with the participants, and after the coding assembly, each group was interviewed on what they thought about the platform. No formal methods were applied on the evaluation.

All groups agreed that the location information was extremely easy to use. Location information was available in a standard Java object, so its usage required no learning. Also, the fact that the virtual world filled itself made the programming easy, as the developers did not need to know in advance what were the names of the areas, but rather, that they existed and they could be used, and the different users in the same area could be easily located.

The developer thought that the main problem area was MUPE XML UI Scripts, and to a lesser extent the bug tracking. The MUPE XML UI Scripts were known to be a problem, as they are something the developers have not used in advance. There is a known learning curve with the scripts, and a portion of the ninety-minute introduction was not enough. During development, most help was required with the scripts, which was reflected in the interviews at the end.
The Bug tracking in MUPE is sometimes tricky, both in Java and XML UI Scripts. The interface for the mobile client needs to satisfy a strict coding convention, and the XML UI Script need to be correctly formulated. These problems lessened towards the end of the code camp, as the developers learned the platform.

3.5 Applications

At the end of the code camp, each group was responsible for presenting their game to the organizers and peer developers. No formal evaluation of the applications was made, as the whole event was squeezed into 24 hours. The organizers also gave a best application award, which was selected based on how well the design had been implemented, and how easy the game UI was.
Bomber and Nibbles managed to design and implement a graphical UI, whereas the other applications relied on a form style UI. All games used the location information in their design and implementation efficiently. All the games implemented their functionality and game logic correctly.
As indicated in the developer comments, the most problems were with the XML UI scripts. This was also evident in the resulting games, where the most problems were with the interaction. A form UI is simple to create, but it is very limited on what it can do.
As a result, the Bomber was awarded the best application award. The deciding factor was the best UI and interaction. Bomber had a good UI, which showed a map of the code camp area, and indicated what was happening and where. The best application award is the WAWC Code Camp itinerant trophy.

3.6 Summary

The programmers thought that location-based applications were easy to develop with MUPE and the extension developed in this work. The problems were with the XML Scripts, and bug tracking, as indicated by both the developer comments and the resulting applications. The code camp itself was also found to be stimulating, as there were a lot of people working on the same problem.

4. CONCLUSIONS

This paper presented how the MUPE application platform can be extended for context-aware applications, and how context-awareness from the server side can be added to the platform. The paper presented two example context producer implementations that simultaneously provided location information. This information was customized for each application with context scripts that mapped user identifiers and made the method calls to the MUPE virtual world based on the context data. The virtual world created its own structure while the players explored the real world, as the system takes new location information as input to fill the virtual world.

With this setup, the world structure, and the location technology are handled by the application platform, allowing the development process to concentrate on the content of the virtual world, the interaction between the players and the actual objects inside the world.

The location-aware MUPE framework was used in a 24-hour non-stop coding session, where location-based games and applications were created. At the start, the participants had no prior knowledge of MUPE, but managed to finish their location based game or application in the given time frame. The participants found the location-based information to be easy to handle with the platform, but they thought that the introduction to MUPE XML UI Scripts should have been longer.

The location-based MUPE virtual world presented here, speeds up the development process of location-based applications in many ways. The location-aware technology is set for all applications after a single setup. The application developers need to only extend one component, the MUPE virtual world. Mobile devices never need to install new software, as the server defines UI and functionality.

ACKNOWLEDGEMENTS

The authors would like to thank Jouka Mattila (Nokia), Timo Nummenmaa (TAY), Janne Oksanen (LUT) for their help in organizing the code camp and WLPR project for location information. Parts of this work have

been funded by AMPERS, and Nomadic Media ITEA project.

REFERENCES

1. Suomela, R., Räsänen, E., Koivisto, A., Mattila, J. and Koskinen, T. 2003, Multi-User Publishing Environment (MUPE) Application Platform. Available from: http://www.mupe.net
2. Dey, A.K., 2000, "Providing Architectural Support for Building Context-Aware Applications, Ph.D. thesis" College of Computing, Georgia Inst itute of Technology, Available from: http://www.cc.gatech.edu/fce/ctk/pubs/dey-thesis.pdf
3. EU-project ePerSpace, 2005. Available online at: http://www.ist-eperspace.org
4. Hawick, K.A., and James, H.A., 2003, Proc. of Workshop of Wearable, Invisible, Context-Aware, Ambient, Pervasive and Ubiquituous Computing, Australian Computer Society
5. Kolari, J., Laakko, T., Hiltunen, T., Ikonen, V., Kulju, M., Suihkonen, R., Toivonen, S. and Virtanen, T., 2004, "Context-Aware Services for Mobile Users (Technology and User Experiences)", VTT Publications 539, ISBN: 951-38-6397-2
6. Tamminen, S., Oulasvirta, A., Toiskallio, K. and Kankainen, A., 2004, Personal and Ubiquituous Computing, 8(2), 135-143, ISSN (on-line): 1617-4917
7. Di Flora, C., Riva, O., Raatikainen, K. and Russo, S., 2004, Poster Proceedings of 6th International UbiComp Conference
8. Korpipää, P., Mäntyjärvi, J., Kela, J., Keränen, H. and Malm, E-J., 2003, IEEE Pervasive Computing, 2(3), 42-51, Volume 2, Issue 3, ISSN (print): 1536-1268
9. Hightower, J., Borriello, G., 2001, Computer ,34(8), 57-66
10. Suomela, R., Lehikoinen, J.: Virtual Reality, 2004, ISSN 1434-9957
11. Geocaching. Available online at: http://www.geocaching.com/
12. Björk, S., Falk, J., Hansson, R., Ljungstrand, P., 2001, INTERACT'01, 423-430
13. Thomas, B., Close, B., Donoghue, J., Squires, J., De Bondi, P., Morris, M., Piekarski, W, 2000, Proc. of the Fourth International Symposium on Wearable Computers, 139-146
14. It's Alive, BotFighters, 2001, Available online at: http://www.botfighters.com/
15. Newt Games, Mogi, 2003 Available online at: http://www.mogimogi.com, 2003.
16. Blast Theory, Can You See Me Now?, 2001, Available online at: http://www.canyouseemenow.co.uk/
17. Sotamaa, 2002, O. Proc. of the Computer Games and Digital Cultures Conference, Tampere University Press, 35-44
18. Nokia Open Source License Version 1.0a (NOKOS License). Available from: http://www.opensource.org/licenses/nokia.php
19. Suomela R., Räsänen E., Koivisto A., and Mattila J., 2004, Proc. of the ICEC, 308 - 320, ISSN: 0302-9743.
20. Multiple User Dimension/Dungeon/Domain (MUD). Available at http://www.moo.mud.org/
21. Luqi, L., and Steigerwald, R., 1992, Proc. of the Twenty-Fifth Hawaii International Conference on System Sciences, 2(7), 470-479
22. Lakkala, H., Context Exchange Protocol Specification, 2003, Available online at: http://www.mupe.net/
23. Lakkala, H. Context Script Specification, 2003, Available online at: http://www.mupe.net/
24. Ekahau web site, Available online at: http://www.ekahau.com
25. WLPR.NET project web site, Available online at: http://www.wlpr.net.
26. 2nd Workshop on Applications of Wireless Communications (WAWC'04) web site, 2004, Available online at: http://www.it.lut.fi/wawc/WAWC04/
27. Results from the WAWC 2004 Code Camp, 2004, Available online at: http://www.it.lut.fi/WAWC/WAWC04/?content=codecamp.html