Real Time Mixed Reality in Virtual Environments

T. Sturgeon, A. Miller, K. Getchell and C. Allison
School of Computer Science, The University of St Andrews
North Haugh, St Andrews, FIFE KY16 9SX
Email: {tommy, alanr, kg, colin}@dcs.st-andrews.ac.uk

Abstract— **This paper addresses the integration of sensor networks, game engines and network protocol technologies. An architecture is presented which will enable these technologies to be leveraged to create mixed reality systems that support a number of applications.**

An instantiation of this architecture has been created. The design and implementation of the system gives a virtual representation of a smart building. The system allows data from sensor networks within the building at known locations to be integrated into a 3D virtual world, facilitating easy monitoring and control of the environment.

This project will provide a solid future test bed for exploring and experimenting with a gateway enabling sensor networks to feed information into game protocols or any other appropriate information rendering platform.

This paper also discusses the challenges that exist in developing such a system.

***Index Terms*—Sensor network, protocols, mixed reality, networking in computer games, computer games, modelling, virtual worlds.**

I. Introduction

This paper addresses the integration of sensor networks, game engines and network protocol technologies. An architecture is presented which will enable these technologies to be leveraged to create mixed reality systems that support a number of applications. The architecture integrates three classes of communication protocols: Internet, sensor network and game engine protocols. Each of these classes has a very different set of requirements. The strict efficiency requirement demanded by sensor network protocols due to their operation in a highly power-constrained environment is not shared by game engine protocols. Game engine protocols instead require frequent data updates and a variety of smoothing techniques to ensure that an enjoyable and fast flowing gaming environment can be presented to players.

The possible ubiquitous nature of sensor networks has been discussed for some time. As the price of each node drops and the processing power increases, cheap mass production is not far off. However processing power has never been the limiting factor for sensor networks, the limiting factor remains the battery power required for continued operation. Whilst there have been many advances in processing power the advances in battery technology have failed to keep up. Due to the constrained amount of power available it is necessary to design sensor network protocols to ensure that they achieve the maximum efficiency possible for their specific application.

The game industry is rapidly expanding and interest in using game technology for educational and training purposes is on the increase. In a mixed reality project it is possible to expand game technology to allow for a more immersive and engaging experience based on real world inputs, thus allowing the development of educations tools and resources more closely aligned to the aims of a constructivist pedagogical approach.

The aim of this project is to create a generic architecture for streaming sensor network data into a game protocol. Once designed a real system will be built using the described methods and architecture in this paper.

An interesting application could be in a command and control centre. A sensor network deployed on the ground could be feeding continuous analysed data to displays in the control room. The sensors may report personnel movement, rapidly changing weather conditions or audio data. This live data and virtual representation could help when making tactical decisions.

II. Architecture

This section provides a high level overview of the way in which information flows through the mixed reality system discussed in this paper before outlining the characteristics and functionality of each of the components which make up the architecture of the system.

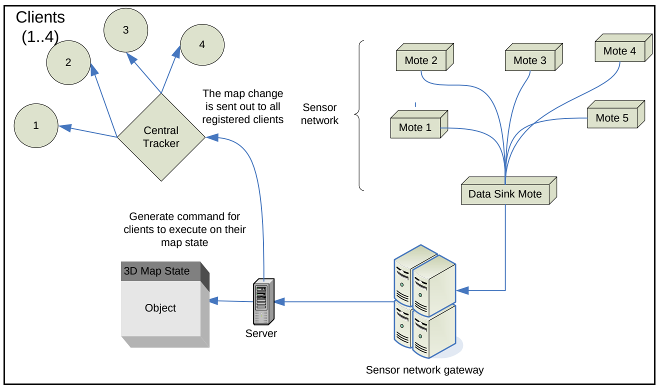

Fig. 1 – Developed Architectural – Component Level

A. *System Component Overview*

This section describes the architecture developed which is capable of supporting the translation of sensor network data into a gaming protocol, thus allowing the sensor network data to modify the virtual environment maintained by the game server. As shown in figure 1, the system comprises of a sensor network, a gateway for passing data between the sensor network and the game environment and a game server which maintains a master version of the current game state and which is able to replicate the current game state to all clients in the virtual environment via a central tracker object.

- ***Mote*** – The mote is a wireless sensor that senses its environment and sends information back to a data sink in the sensor network. The mote is constrained by battery power and as such data cannot be sent with high regularity or with large redundancy.
- ***Data Sink*** – A data sink in the sensor network is a central point where data from motes is collected. The motes may send their data directly to the data sink, or the data may be routed via other motes in the sensor network before arriving at the data sink.
- ***Sensor Network Gateway*** – The gateway provides a means of extracting data from the protocols used by the sensor network and inserting it into protocols understood by the game engine maintaining the representation of the virtual world. The gateway needs to be capable of a fast and efficient translation process in order to ensure that the real world data enters the virtual world representation whilst it is still relevant and valid.
- ***Central Tracker*** – The central tracker ensures that any allowable updates to the virtual world triggered by game clients are replicated amongst all game clients in the same virtual world. The central tracker is also responsible for sending map updates and other control commands, e.g. increasing light intensity in a certain area, as dictated by the game engine server to all game clients. The central tracker is capable of differentiating between clients joining different virtual worlds, new clients joining who need full map updates and current clients who are already actively engaged in a virtual world and who need a replica of another client's update.
- ***Game Server*** – The game server is the central component that regulates readings from the sensor network, assigns virtual world actions based on that data and maintains the state of the virtual world state. The game server also contains a listing of all clients, their locations within the virtual world and their individual states. Below is a list outlining some of the operations that the game server performs:
    - Maintains and Checks for timeouts from clients and removes any client where the timeout has expired from the central tracker.
    - Gathers information from the sensor network and displays it within the virtual world in an appropriate fashion.
    - Authorises clients joining and updating virtual world state.
    - Maintains a security policy to ensure that all client interactions with the virtual world are applied as per the game rules.
- ***Clients*** - The clients each run their own 3D engine that displays to the user the current state of the virtual world. Clients must register at the start of a session and updates to their representation of the virtual world are handled by the central tracker. In order to ensure a smooth, fast flowing virtual environment the emphasis on client communication is efficiency and speed with lightweight UDP communication being used to provide rapid updates to the world state maintained by each client. More reliable (and less efficient) TCP communication between the clients and the central tracker is used sparingly to ensure important control information is accurately replicated amongst participants.
- ***Map*** – The map is a virtual representation of a real world location. The map can be of anything so long as it is accurately represented, i.e. the real world position of the sensors must match the position in the virtual world. When generating the map one can use architectural schematics to maintain the correct dimensions. The environment represented by the map can be anything from a beach to a hotel. In this paper the School of Computer Science was chosen as the building to model from architectural plans.

B. *High Level Data Flow and Interaction*

As shown in figure 2, readings from the sensor network motes distributed throughout the real world are sent to the virtual world through the sensor network gateway. The sensed data is then interpreted at the game server and, if applicable, a modification to the virtual world maintained by the server is applied. The virtual world modification is then replicated amongst all registered clients by the central tracker.

A client will frequently send an update to the virtual world

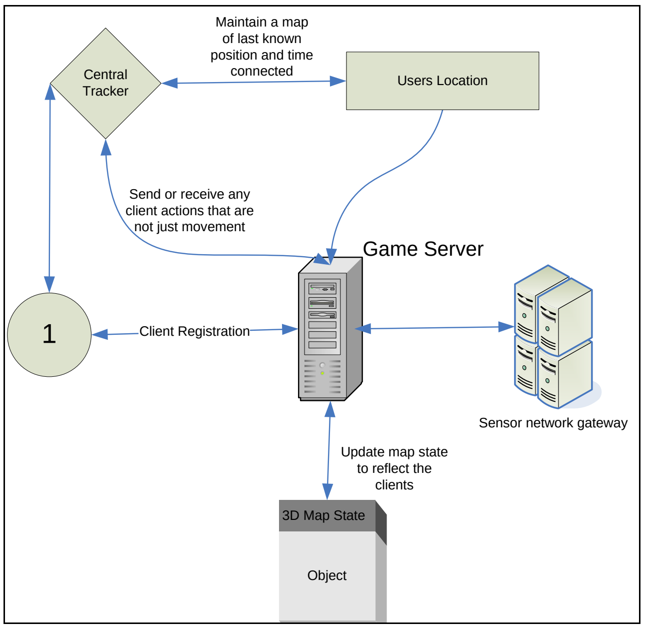
Maintain a map of last known position and time connected
Central Tracker
Users Location
Send or receive any client actions that are not just movement
Game Server
1
Client Registration
Sensor network gateway
Update map state to reflect the clients
3D Map State
Object

Fig. 2 – Developed Architecture - Overview

state maintained by the server via the central tracker. Assuming that this update can be validated as possible by the server, the centralised tracker will then replicate this change in virtual world state to all known clients.

At present the system is capable of displaying the temperature and light intensity sensed in an area of the map. Since the School of Computer Science has motion sensors in each room which turn on the lights when motion is detected, modifications are being developed which will allow the virtual world representation to display rooms in which people (or at least) movement are present by interacting with the building control systems. Another expansion which is currently being explored is to develop a thermal image of the building, thus allowing heat sources, such as radiators, people etc to be displayed in the virtual environment.

C. *Instantiation*

The above architecture has been used in the implementation of a mixed reality system. The developed system integrates live data from a sensor network inside the department into a virtual environment. The rest of this paper discusses the design issues, technologies and challenges that were encountered during the instantiation of the above architecture. In particular, the 2D to 3D conversion necessary to create an accurate model of the real world is discussed along with the technologies used to render the 3D world, issue in network code design and sensor networks.

III. Developing A Virtual World Map

A. *Platform Choice*

There are many design considerations when modelling a building in a virtual world. One option was to build the virtual world using an existing game engine as the basis for development. Various engines were investigated, such as the Quake Engine from ID, the Unreal Runtime from Unreal and the Source Engine from Valve and eventually chose to focus on the Quake engine as it had been recently open sourced, thus allowing us the opportunity to modify the underlying engine itself if necessary. The initial maps were built using a mixture of the GTKRadiant[1] and Q3Radiant tools. However it quickly became apparent that developing a realistic virtual representation of a real world environment was going to be problematic using this platform. Without any point of reference it was difficult to scale the environment correctly and so a short experiment was undertaken in order to determine how users' perceived the relative scale of items in the virtual environment. The experimental phase provided inconclusive results, with users' perceptions of size varying greatly. As no consensus was met, it was decided to reassess the initial decision to base the environment on a standard game engine. Instead a customised engine was developed. This allowed the virtual map to be based on accurately scaled architectural plans.

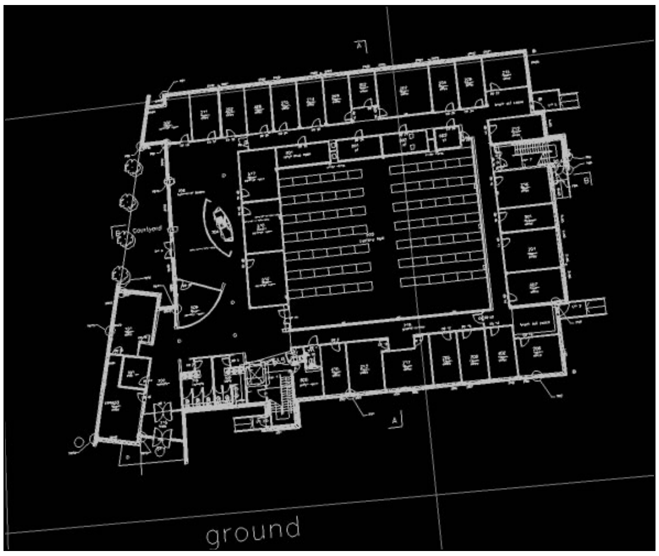

Fig. 3 – Original 2D Architectural Plans of the Building to be Modeled Virtually

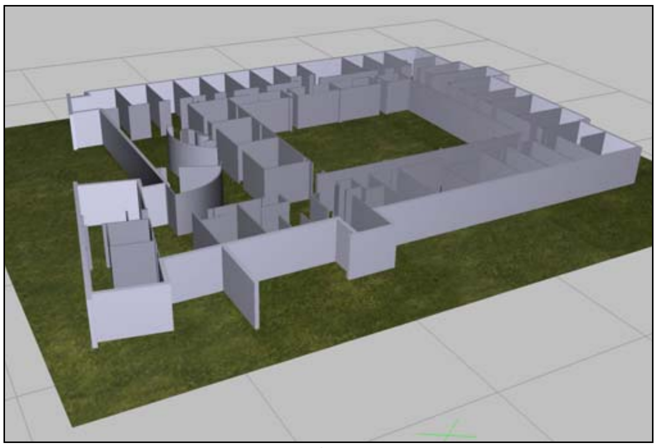

Fig. 4 – Finalised 3D Model of Building based on Original 2D Architectural Plans

B. *Building a 3D Model from 2D Architectural Plans*

The architectural plans for the School of Computer Science building were provided in 2D Autodesk® AutoCAD® DWG format. As can be seen in figure 3, the original architectural plans contained a great deal of information regarding the positioning of building elements. Some of this information, for example the positioning of doors, windows, internal walls etc was required when developing the 3D model whilst other sections of the original drawing were not required, for example the positioning of drainage, cable conduits, furniture etc. In order to simplify the conversion process this extraneous information was removed in Autodesk® AutoCAD®. Once the 2D plans had been modified to contain only the information required to develop the 3D model this data was imported into Autodesk® 3DS Max® 8.

The process of developing the 3D model was documented fully during the process {ref} and broadly follows the strategy adopted by [2]. To briefly summaries, the walls were converted to closed splines and extruded to give the 3D shape of the building. After extruding the walls a UVW map was applied to enable texturing of the finished structure. Once the underlying 3D model was completed it was exported to the Autodesk® 3DS Max® 8 3DS file format so that it could be imported into both the texturing package, Right Hemisphere Deep Exploration, and also into the 3D engine used by the clients and server whilst rendering the virtual world map.

As shown in figure 4, the finished model accurately portrays the original 2D architectural plans in terms of both the scale of the elements which make up the building and also their positioning within the map area.

IV. 3D ENGINE

The 3D engine is required to display the virtual world to the client. As discussed in section III, there are various 3D engines available which can be used to model a virtual environment. Commercial products such as the free Unreal Runtime Engine or the purchasable Valve Source engine are limited in their ability to be modified due to the restrictions that their licensing imposes. Open source engines such as the ID Quake engine provide a more adaptable environment owing to the ability to modify the operation of the engine at the source code level. Both the commercial and open source engines discussed all share one thing in common; the need for the client to download and install the engine itself.

Given the scale and perception problems experienced in the early stages of working with pre-existing game engines, it was decided that a customised engine should be developed. A key consideration was to make it possible to quickly distribute the game engine to each client over the World Wide Web. This would allow anytime-anywhere access to the system without requiring clients to use any bespoke software. By using Java it was possible to make use of existing Java 3D engines able to render 3D models formatted using Autodesk® 3DS Max® 8 3DS files whilst also allowing the entire engine to be compressed into an easily portable JAR file which can be distributed to each client using Java Network Launching Protocol (JNLP) technology.

The Java 3D engine used to render the virtual environment is jPCT[13]. jPCT supports software and hardware rendering with OpenGL support. A clear advantage of using jPCT is the open source nature of the project, unlike most commercial game engines with jPCT it is possible to directly modify the source code. This level of flexibility would not be possible if a commercial game engine had been used.

V. NETWORK CODE DESIGN

For networked games the client updates the server with its current state approximately 30 times a second. The client is given the entire map state on initial connection then supplied with updates indicating a change to a certain area of a map. These updates to the map state are broadcast to all clients every x seconds. The value of x varies depending on how fast the server is. In network games such as Valve Counter-Strike:Source various techniques are employed to compensate for users with a higher latency connection to the server [3] :

- ***Delta Compression*:** This feature involves not sending the state of the game in its entirety every update period but rather sending information on what has changed only.
- ***Entity Interpolation*:** When a client is receiving information about other players movements it does so at distinct intervals of around 30 times a second. If the software was to draw the movements of a player as 30 distinct movements the result is a stuttering effect. This stuttering is avoided by delaying all packets by 100ms and filling in any missing or corrupted data to give a smooth animation.
- ***Input Prediction*:** This involves each client running the same calculation code as the server to give the user immediate feedback as to their position and status rather than waiting for server confirmation.
- ***Lag Compensation*:** Whilst a client might see another client at time x the other player may have already moved away, the server remembers previous positions of players for a given length of time. When a client sends an event packet the server logs are checked for the system state at the time the client initially made the request.

The architecture must be capable of displaying one user's actions to another and incorporating any real time server data. Each user needs to know every other user's position, orientation and state.

A main design decision was the choice between having a central point that all data must flow through and creating a P2P system for exchanging data for the exchange of data between clients and server. P2P offers no central point of failure however the centralised server does have some key advantages especially when considering future expansions possibilities. The centralised server allows for clients with higher latency since the only update they have to send is to the server.

The advantage of the configuration illustrated in figure 5 is that the nodes that have the faster links on the network will do most of the data transfer. Now node 4 only has to update one other node rather than having to notify all other members of the group directly. The above configuration will reorganise itself in case of a node leaving the group.

The centralised server, as shown in figure 6, also offers key security controls, as having a certified central server can insure against an attacker feeding false sensor data to the clients. Another advantage is in the prevention of cheating by any of the nodes. The anti-cheating policy being that important computations and decisions are controlled by the server and the nodes notified of significant changes.

The current server implemented uses a central tracker to relay information from one client to all other clients registered as having the same session identifier. However in a P2P group a set of central-like servers could be elected. These

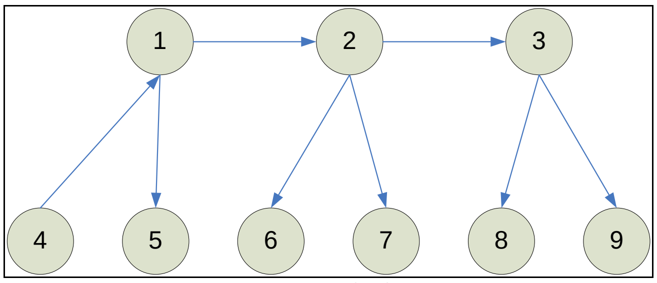

Fig. 5 – Diagram Showing the Flow of Information in a P2P Network with Elected Leaders 1, 2 and 3.

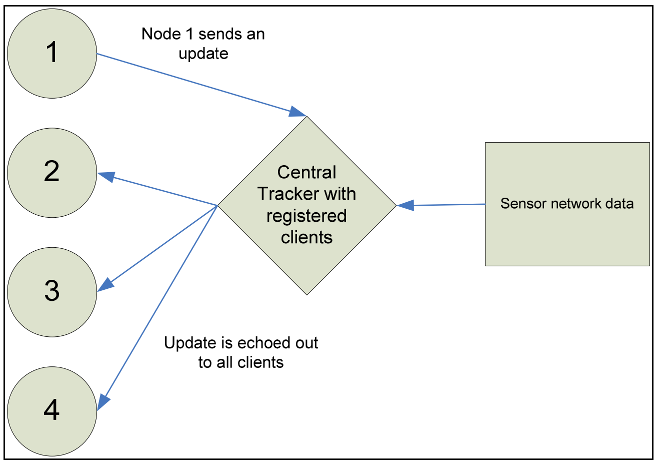
1
Node 1 sends an update
2
Central Tracker with registered clients
Sensor network data
3
Update is echoed out to all clients
4

Fig. 6 – Configuration of a Centralised Way of Transmitting Data to All Clients Involved.

master nodes would then transmit the information to members of its group.

VI. Sensor Network and Translation Gateway

Initially there will be around 10 sensor nodes situated in the building. These nodes will be running TinyOS and have been programmed to report on their environment every 30 seconds. The data sensed by the node is returned to the data sink. If a node is outside the range of the data sink then the P2P nature of the network will route the message to the data sink. Once at the data sink node, the data is retrieved from the node by a serial port connection. The data read from the serial port is then printed to a standard net socket. The socket accepts connections and will simply return the data read from nodes in real time. Since the data from a node is relevant to its physical location determining a nodes location is important. Initially each node can be named and its position in the building noted during installation however this method will not support nodes moving around.

There are various methods for calculating a nodes location such as GPS enabled nodes however GPS does not work indoors. A future expansion will be to have a few nodes with known locations. Then the relative distance between a known node and unknown node can be calculated and the absolute position determined.

VII. Testing

A high level packet modifier for UDP packets was written to test the system. The packet modifier will corrupt data, reorder packets, delay packets or drop the packet. A user interface was developed to allow the user complete control over how many packets to interfere with and for how long. Initial tests for dropping around 20 packets (s^{-1}) or dropping a large amount of packets randomly caused a jittery display of other users' movements and highlighted the need for entity interpolation.

Simple error checking avoided any problems with data scrambling and a highest received packet counter removed any problems with reordering of packets.

A disadvantage to the packet modifier written is that it operates at an application level rather than at the kernel level and as such will not perform optimally.

VIII. Related Work

Using the virtual world to portray what the real world will look like, for example once a new building has been completed, has been used by many architects [2]. The approach used in [2] is particularly relevant because of the use of architectural plans to create the 3D environment. However [2] uses the commercial game engine Unreal which therefore requires all end-users to have paid for and installed the Unreal engine. The advantage of this project is the use of a Java open source 3D engine which provides portability to anyone with a Java Runtime Environment installed and requires no installation or payment for any game applications.

The system created in [4] allowed for players online to be chased by runners on the streets of a city. This project incorporates the real time position of a person using the GPS system and can relate this to online players over the Internet. A major problem with [4] was the accuracy of GPS in built up areas. The accuracy of GPS degrades significantly around high compacted buildings and stops working completely if indoors. However the launch of the new European satellite navigation system [5] will provide greatly improved accuracy. Whilst [4] provides information on the runners locations in the city, no real time information is provided about the conditions in the city. One could imagine that by integrating a sensor network that detects size of flow of people in a city street then that information could dramatically change the game play. The augmentation of real information into a virtual world with efficiency and accuracy is what this paper is attempting to create.

A similar project to [4] was ARQuake [6] in which a head mounted display (HMD) is used to show the wearer the current game state in the virtual world whilst they are walking around the real world. This augmented reality approach provides a very powerful and immersive experience however the equipment needed to play such a game is expensive. One advantage this project has it that all that is needed to interact with the system is a Java runtime environment. However the obvious drawback being that such a display is not as immersive as a HMD. Mixed reality systems are increasingly being researched [7-12] and producing innovative and technologically challenging systems.

IX. Future Work

The system is currently a work in progress. A full implementation of the architecture outlined in this paper is very close. Future expansion areas being considered are the tracking of personnel in the building through either motion detecting or individual tracking motes. Another desirable expansion is to allow the virtual world to cause an action in the real world; early examples include, but not limited to, the control of light switches, kettles, doors or coffee machines.

X. Conclusion

Mixed reality projects involve the use of many different technologies in a new and challenging way. Research into combining sensor networks and game technologies has been sparse at best. This paper has outlined a suitable architecture for creating an environment where sensor networks and a virtual environment can exchange information. The end system will provide a solid test bed for future research and development in various areas of sensor network and game protocols.

References

[1] GTKRadiant, id Software, http://www.qeradiant.com/

[2] Shiratuddin, M.F. and Thabet, W. (2002) Virtual Office Walkthrough Using a 3D Game Engine, in *International Journal of Design Computing, 4*, http://www.arch.usyd.edu.au/kcdc/journal/vol4/.

[3] Valve Corporation (2004). "Source Multiplayer Networking" (URL) http://www.valve-erc.com/srcsdk/general/multiplayer_networking.html

[4] Flintham, Anastasi, Benford, Hemmings, Crabtree (2003). Where On-Line Meets On-The-Streets: Experiences With Mobile Mixed Reality Games.

[4] Bjork, S., Falk, J., Hansson, R., Ljungstrand, P. (2001). "Pirates! Using the Physical World as a Game Board". Interact 2001.

[5] Galileo , European Satellite Navigation System, http://europa.eu.int/comm/dgs/energy_transport/galileo/index_en.htm

[6] Thomas, B. and Piekarski, W., "ARQuake: The Outdoor Augmented Reality Gaming System", Communications of the ACM,2002 Vol 45. No 1, pp 36-38

[7] Bannon, Benford, Bowers, Heath (2005). Hybrid design creates innovative museum experiences. Bjork, S., Falk, J., Hansson, R., Ljungstrand, P. (2001).

[8] Benford, Magerkurth. (2005). Bridging the physical and digital in pervasive gaming. Bjork, S., Falk, J., Hansson, R., Ljungstrand, P. (2001).

[9] Prince, Cheok, Farbiz. 3D Live: Real Time Captured Content for Mixed Reality.

[10] Benford, S. et al. Bridging the Physical and Digital in Pervasive Gaming. Communications of the ACM. Vol. 48, issue 3 (2005) 54–57.

[11] Mitchell M. et al. Six in the city: Introducing Real Tournament—a mobile IPv6 based context-aware multiplayer game. In Proc. ACM NETGAMES '03 (2nd workshop on network and system support for games) (2003) 91–100.

[12] Fourth IEEE and ACM International Symposium on Mixed and Augmented Reality (2005) in Arlington, VA, USA (URL) http://campar.in.tum.de/ISMAR/WebHome

[13] jCPT, http://www.jpct.net/