Toward a Mobile Platform for Pervasive Games

Inseok Hwang*, Youngki Lee*, Taiwoo Park*, Junehwa Song
Dept. of Computer Science, Korea Advanced Institute of Science and Technology
Daejeon, Republic of Korea
{ inseok | youngki | twpark | junesong }@nclab.kaist.ac.kr
* The first three authors are listed in alphabetical order.

ABSTRACT

Emerging *pervasive games* will be immersed into real-life situations and leverage new types of contextual interactions therein. For instance, a player's punching gesture, running activity, and fast heart rate conditions can be used as the game inputs. Although the contextual interaction is the core building blocks of pervasive games, individual game developers hardly utilize a rich set of interactions within a game play. Most challenging, it is significantly difficult for developers to expect dynamic availability of input devices in real life, and adapt to the situation without system-level support. Also, it is challenging to coordinate its resource use with other gaming logics or applications. To address such challenges, we propose *Player Space Director* (PSD), a novel mobile platform for pervasive games. PSD facilitates the game developers to incorporate diverse contextual interactions in their game without considering complications in player's real-life situations, e.g., heterogeneity, dynamics or resource scarcity of input devices. We implemented the PSD prototype on mobile devices, diverse set of sensors, and actuators. On top of PSD, we developed three exploratory applications, *ULifeAvatar*, *Swan Boat*, *U-Theater,* and showed the effectiveness of PSD through extensive deployment of those games.

Categories and Subject Descriptors

C.3 [**Computer Systems Organization**]: Special-Purpose and Application-Based Systems; C.5.3 [**Computer System Implementation**]: Microcomputers – *Portable devices*; K.8.0 [**Personal Computing**]: General – *Games.*

General Terms

Design, Experimentation, Human Factors.

Keywords

Pervasive Game, Platform, Mobile

1. INTRODUCTION

We are at the dawn of making the pervasive computing true reality. Following today's wide penetration of mobile devices such as smartphones, subsequent proliferation of advanced wearable devices and sensors will give rich digital interactivity on our surrounding environments, our daily activities, our social interactions, and so on. And we believe that, it will hugely expand the ways of interaction between us and what we call games today, building the concept of *pervasive games*.

We envision that the pervasive games will be mostly distinguished by the immersion into every facet of our real lives. Getting out of the virtual world inside a small screen with dedicated controllers, a pervasive game will be running 24 hours a day, integrating a player's every gesture, every footstep, every items around him, and even every place he visits into a part of the game. We imagine a pervasive version of MMORPG (massively multiplayer online role-playing game), where the player develops and strengthens his avatar through his everyday real world interactions. Every stair he climbs, his avatar becomes stronger. Every time he eats fresh vegetable, his avatar regains stamina. Every new place he visits, his avatar explores a new place. Every time he breathes dusty air, his avatar loses a few hit points (HP). Every new person he shakes hands with, their avatars becomes friends to each other, and so on.

Developing such pervasive games closely integrated with the player's real life introduces a number of challenges and considerations. First, the game requires high level semantic definitions of the player's behaviors to be used as game interactions, e.g. climbing stairs or shaking hands. Second, the game should be highly flexible in the way to monitor and classify such input semantics. Hardcoding the device types would harm the ubiquity of the game for many players with different device availability in different surrounding environments. For example, climbing stairs can be detected by monitoring the player's shoe-embedded accelerometers, or by utilizing the building's indoor localization infrastructure, or maybe by monitoring the altitude from the barometer built in his smartphone. Third, the underlying monitoring process should be energy-aware and situation-adaptive. The game would not be the sole application which can monopolize all the devices and resources. Continuous use of smartphone-embedded sensors would rapidly exhaust the smartphone's battery, limiting the player's usage of his phone for the remaining hours of the day before coming back home. Switching to using internal localizations where available would be energy-efficient alternative.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
MobiGames'12, August 13, 2012, Helsinki, Finland.
Copyright 2012 ACM 978-1-4503-1487-9/12/08... $15.00.

In this paper, we propose our initial design of Player Space Director (PSD), a mobile platform for developing and running various pervasive games. PSD effectively provides the game designer with the definition of diverse contextual interactions, fully reflecting player's real-life situations. In runtime, PSD orchestrates the interactions between a player and game, i.e., the "player's space". More specifically, PSD primarily targets:

- To interpret pervasive game interactions, recognizing player's gestures, actions, and surrounding contexts and abstracting them as a meaningful game state and an input.
- To abstract and manage a large number of heterogeneous sensors/actuators in the player's pervasive game spaces.
- To keep track of the resource availability around the game space, and coordinate the resource usage accordingly.
- To communicate with the game-specific logic running on top of PSD.
- To coordinate player- / space-heterogeneities to enable in-situ or remote multi plays among two or more PSD-supported game players.

Figure 1 illustrates the top-level logical view of a pervasive game with PSD. A pervasive game space is composed of a game logic and a number of gamers interacting with the game logic. The gamers interact with the game in their local space which is enhanced with a number of sensors and actuators. The gamers' actions and gestures are interpreted and used as meaningful gaming input, along with the environmental situations around the gaming space.

The rest of this paper is organized as follows. Section 2 introduces the early designs of pervasive games and discusses the efforts to support pervasive games. Section 3 proposes the initial architecture of PSD and its major components. In section 4, we present the practical use cases of PSD by showing our pilot implementation of pervasive games on top of PSD. Then, we conclude this paper with projecting the further evolution path of pervasive games.

2. RELATED WORKS

In this section, we introduce diverse pervasive game designs that incorporate various types of interactions or real life situations. Also, we present early platforms for pervasive games and differentiate them with PSD.

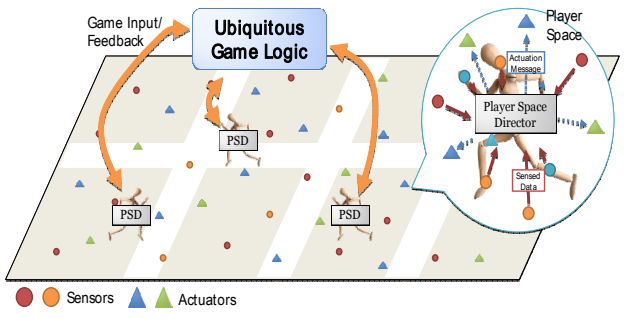

Figure 1. General model for a pervasive game

2.1 Pervasive Games

Pervasive games enrich the player-to-game interactions far more than those of conventional games, by harnessing a variety of sensors devices. For instance, Wii Sports [19] introduces gesture-based, more intuitive game plays. A player grips Wii mote, embedding accelerometers and gyroscopes to estimate the motion of the player's arm or wrist. Pirate! [4] is a proximity-based multiplayer game, utilizing the distances between players estimated by RF-based proximity sensors. Pulse Masters Biathlon [17] uses the player's heart rate as the primary interaction. The player wearing a heartbeat sensor does whatever to change his heart rate, which then controls an avatar playing biathlon.

As pervasive games extend the way of interactions, the integration of games and real life is being expedited. REXplorer [1] is designed for tourists to play at major tourist attractions. Tourists are given a "magic wand", a camera-equipped smartphone. At a historic site, they are asked to cast a site-specific spell, i.e. a phone-recognizable gesture. Then the phone narrates a historic story and guides them for further progress. UbiFit Garden [6] and Fish'n'Steps [12] uses gamification to promote daily exercises. The player's physical activities are sensed and make the virtual garden blossom, or feed the virtual fish to grow, respectively. Similarly, Musolesi et al., proposed a Second Life-like virtual world which reflects users' daily life behavior [17].

Playful Toothbrush [5] is an educational game for preschoolers, designed to arouse children's interest and help them to develop proper brushing habits.

2.2 Platforms for Pervasive Games

While pervasive games have been receiving growing attention of research community, underlying platforms for pervasive games have been little pioneered yet. Many efforts have been focusing on genre-specific platforms, such as augmented realities, tabletop, and location-based games [7][10][15][16], rather than generic platforms. Also, general context-aware application platforms have been proposed [12][13][20], however, they do not mainly aim to support diverse characteristics of pervasive games. The pervasive game examples in Section 2.1 are best featured by context-based interactions and the player-mobility to support real world game play. A generic platform for pervasive games would be required to effectively support those features, assisting easy and rapid development.

An early framework for pervasive games was proposed in [15], aiming at context-aware provisioning of multimedia contents for mobile game players. A key difference of its design from PSD is that it utilizes a server to manage the context information. Such server-centric supervision of the player contexts, however, may raise the privacy concerns and security issues. Besides, the server would be heavily burdened with the context processing, while the client

devices mostly remain as simple viewers. In addition, the transmission of voluminous sensor data would quickly drain the clients' battery. PSD, on the other hand, operates on the player's device and directly processes player's contexts. It prevents many possible privacy breaches. The game server, if any, is free from low-level context processing, and just subscribe for events needed for multi-player coordination. The overall communication can be greatly saved as well.

In terms of interactions, STARS [16] targets computer-augmented board games where multiple input or output modes are available. STARS aids game developers to choose effective means of interactions by evaluating proper combination of interaction modes fitting to the developer's criteria. PSD provides the developers with 'player context vocabulary', which predefines commonly used interactions in pervasive games. However, PSD provides much richer vocabulary set covering diverse interactions with extensive devices to support diverse real world activities. Note that STARS only targets screen-based interactions in PDAs.

3. PLAYER SPACE DIRECTOR

PSD is a mobile game platform to support contextual interaction and diverse input modalities and thereby facilitate the development and operation of diverse pervasive games. It supports game designers to define a rich set of pervasive game interactions for their own use. Such defined pervasive game interactions are automatically interpreted and processed by PSD runtime environment. The PSD platform also orchestrates player spaces, abstracting and managing diverse heterogeneous devices in pervasive game spaces.

3.1 PSD API

PSD provides game developers with simple and intuitive API, to facilitate the specification of contextual inputs. *registerCIQ()* is the key API. Using *registerCIQ()*, applications specify contextual inputs to use in the game in the form of *Contextual Interaction Query* (CIQ). CIQ is an intuitive query language that supports rich semantics for a wide range of contextual inputs. PSD provides well-abstracted context vocabulary used in CIQ (see Table 1 for whole vocabulary of contextual inputs in PSD) Using CIQ, game developers easily leverage diverse contextual inputs needed for game interactions. They do not have to consider the details of underlying complications in its sensor usage and data processing.

For example, a game developer wants to utilize every user gesture when the user is running. Then, he can simply register a CIQ as follows.

GAMEINPUT gesture
CONDITION (activity==running)
REPORT Immediate

CIQ supports diverse types of result reporting. Currently, we support three types: *Once*, *Periodic*, and *Immediate*. In the *Once* mode, PSD reports a result once only when CIQ is issued. With *Periodic* mode, applications are reported upon every configured time period. If *Immediate* type is used, a result is reported whenever a new result is available.

Table 1. Possible contextual inputs in pervasive games

Player's gesture	Punching, clapping, shaking hands, waving, kicking
Player's activity	Running, walking, sitting, brushing, bicycling
Player's physical condition	Heart rate, blood pressure, sweating, body temperature
Player's surrounding situation	Location, air temperature, wind speed, humidity, loudness

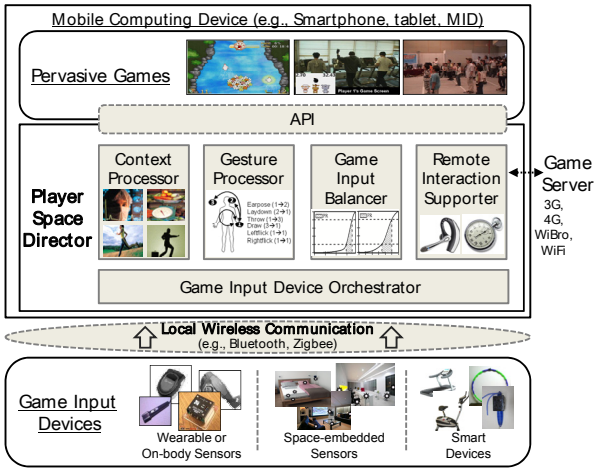

Figure 2. PSD architecture overview

3.2 Context Processor

Context processor of PSD process CIQs that request about player's activity, physical condition and surrounding situations. Note that player's gesture is separately handled due to its unique characteristics. Inside the context processor, PSD incorporates diverse processing modules, which are used to extract a variety of contextual inputs from raw sensor data. For instance, it includes diverse feature extractors such as FFT (Fast Fourier Transform), MFCCs (Mel-frequencly Cepstral Coefficients), and statistics processing modules, to extract summarized features from raw sensing data. Furthermore, it includes diverse classifiers such as decision tree, HMM (Hidden Markov Model), GMM (Gaussian Mixture Model), to extract and classify high level contextual input from summarized feature data.

PSD internally abstracts each processing pipeline to handle a CIQ as a *processing plan*. A *processing plan* is a pipeline of processing tasks and used as a unit of resource allocation. A plan often involves a set of sensing, feature extraction, and context recognition tasks over distributed devices. Specifically, the input device-side processor performs early-stage of the context processing pipeline such as sensing tasks and optional feature extraction tasks. The processor in the mobile device executes the rest, i.e., feature extraction and context recognition tasks or the latter only, and completing plan execution. Note that PSD

prepares multiple processing plans for a contextual input and selectively utilize them according to sensor availability and contentions in sensor resource use.

3.3 Gesture Processor

The *gesture processor* supports hand gesture input which is a natural and promising user interface modality for mobile and pervasive games. It utilizes wearable and on-body motion sensors such as wristwatch-type accelerometers and gyroscopes to obtain users' hand motion data. Unlike conventional keyboard- and touch screen-based interaction, hand gestures can simplify on-the-move interaction by reducing the need to concentrate much on mobile devices. With the support of a flexible and programmable gesture-based game interaction, a user can seamlessly interact with various mobile and pervasive games.

The gesture processor actively copes with two important challenges in developing mobile and pervasive gesture interaction system: users' mobility and limited energy of sensor and mobile devices. Traditional gesture processing pipelines (usually based on HMM) not designed for mobile and pervasive use may quickly drain the batteries of sensor and mobile devices or fail to accurately recognize users' gestural inputs. First, the gesture processor uses sensor-side gesture pre-filtering scheme to save energy for continuous motion data transmission, based on sporadic nature of hand gestural inputs. Second, it supports mobility-resilient gesture recognition by multi-mobility situation-based HMM training scheme. The key idea of the scheme is to train HMM using gesture samples from a set of representative mobility situations.

3.4 Game Input Device Orchestrator

Given the registered CIQs, the *game input device orchestrator* decides how to process the requests with the available input devices and their resources. It includes two major sub-components: the *plan generator* and the *plan selector*. The plan generator dynamically updates applicable plans based on available input devices and their processing capabilities. Among the generated plans, the plan selector decides a set of plans to execute to support registered CIQs. The selection changes adaptively, reflecting dynamic availability of input devices and their resources.

For effective resource use planning, it is critical to acquire the resource information. The resource monitors keep track of available resource status of input devices and a mobile device itself. They continuously monitor the status of CPU, memory, energy, and bandwidth. The status is periodically reported to the plan selector for runtime adaptation. The monitors are designed to minimize monitoring overhead while providing reasonable accuracy.

3.5 Game Input Balancer

The game input balancer is designed to enable pervasive game players to closely synchronize with each other with

Figure 3. PSD prototype hardware settings

different input modalities. Imbalances in gaming performance caused by different input modalities can significantly degrade users' sense of synchronization and fun. For example, we imagine an exercise-based racing game, which offers a large variety of options in sensor-embedded exercise equipment, ranging from treadmills or stationary cycles to a handy jump rope. To enable multi-player games between players with different choices, the game input balancer gives proper advantages or disadvantages for each input modality, based on the characteristics of input modalities such as rapidness and accuracies while changing game values. These balancing schemes should be carefully applied by analyzing core mechanics of given games. To this end, we provide game designers with the characteristics of input modalities and help them to choose appropriate balancing mechanism for core mechanics of their games.

3.6 Remote Interaction Supporter

The remote interaction supporter provides communication mediums such as video and audio channels to facilitate social interactions during game play. It selectively supports various modes of communications on-demand, namely peer-to-peer, team, and all-player communications. It also supports communication between game applications and a game server while effectively suppressing unfairness in game play due to the differences in network delay variation among players.

4. IMPLEMENTATION

We have implemented a PSD prototype, carefully implementing the current PSD architecture introduced in Section 3. Currently, the prototype is implemented in C++ on a Linux and Java on an Android platform. Sensor device implementation is based on NesC over TinyOS.

Deploying PSD requires three important hardware sets: mobile devices, sensors, and actuators. Figure 3 shows a

snapshot of currently used hardware. First, we have deployed the PSD prototype on NexusOne Android phone and an Ultra Mobile PC with 1.33 GHz CPU and 1GB RAM. For sensor nodes, we used multiple KNodes and USS-2400 [11] nodes with diverse sensing modules such as accelerometers, gyroscopes, GPS, environmental sensors, and bio-medical sensors (Blood Volume Pressure sensor and Galvanic Skin Response sensor, ECG sensor). In particular, we attach an acceleration sensor in a glove for hand gesture detection; the glove is easy to wear in real game situation. They are equipped with Atmega 128L MCU, CC2420 RF module supporting 2.4GHz band ZigBee protocol, and TinyOS as operating system.

To create fun pervasive games, we also incorporate diverse types of application-specific actuators. Along with our prototype applications, we developed an application-controllable smart treadmill. It has a built-in socket interface to receive commands from applications, such as acceleration or inclination. In addition, it can detect the runner's over- or under-pace by an infrared sensor measuring the distance from the runner.

From a developer's standpoint, developing pervasive games is quite simpler with PSD; we only need to define the game logics and design user interfaces. Many game interactions are signaled by gestures mimicking real actions, like shaking an arm wearing an accelerometer to toss an item in the game. The complexities in processing accelerometer readings and recognizing the gestures are completely handled by PSD while the game logics can be reduced to a simple CIQ registration on PSD.

5. PROTOTYPE GAMES

5.1 ULifeAvatar

ULifeAvatar is a sample lifestyle pervasive game which generates an avatar reflecting characteristics of players' real lives, as described in Section 1. More specifically, avatars have several attributes such as strength, dexterity, intelligence and charisma, which are common stats in many role playing games. These attributes are gradually affected on avatars by the players' daily activities. For example, the game continuously monitors players' walking and running speed and log the maximum and average speed. Then the logged values are converted into the avatar's average and instant maximum dexterity, which can be used in other pervasive social games. The game also monitors the number of people which players meet daily, accumulates the number day by day, and changes the charisma value for corresponding avatars. In Swan Boat racing game which will be explained later, a player with a higher charisma value has higher possibility to obtain a booster item.

Figure 4. Playing Swan Boat (left) and U-theater (right)

5.2 U-Theater

U-theater is a group-interactive gaming environment for public places with a large screen, like conventional movie theaters. (It has been introduced in our previous work [12]) We suppose typical games in U-theater are likely played by tens of players at the same time as shown in Figure 4. Each player wears sensors, watches the screen, and plays the game with gestures. U-theater would introduce a unique gaming experience of group collaboration or competition involving physical activities. We have developed three game applications for U-theater: "Cheer together!", "Smash the beehive!", and "Jump, Jump!" Here we only elaborate "Smash the beehive!" due to the page limit. A beehive with a lot of bees swarming around is shown on the screen. When the bees are swarming, the players should freeze not to get stung by the bees. Then, suddenly there comes a moment when no bees fly. The players should take that chance to punch the beehive. The quickest gets points. The amount of points is proportional to the strength value of the avatar of the quickest player. If more than two players hit the beehive at the same time, the player which has an avatar with higher dexterity value wins.

To study end-user experiences, we held a public event to demonstrate U-theater. 32 elementary students within 3rd through 6th grade participated in the event. It was a single day event, and they were divided into two groups with 16 players each. To instantly generate avatars' attribute values, we gave players 100 points and ask them to distribute among attributes, and used the answers as the attribute values for their avatars. Each group experienced all three U-theater games we developed. We have observed and analyzed their behaviors both on-site and from video recordings.

Prior to the event, we expected that the unique fun factors obtainable from PSD-based games were two-fold: the gesture-based game plays and the group gaming experience. From these standpoints, the players have shown behaviors closely concurring with our expectation. Moreover, a notable discovery is that both fun factors are not just valid but also highly correlated. To be specific, the gestures, which are the prime modes of interaction in PSD-based games, essentially enable active social interaction. As the game interactions are merged with into natural motions, our games give more freedom of motions than classical games. Particularly, relieving eyes and limbs of excessive game controls enables intensive interaction with neighbor players involving eye contacts and physical signals. Even though those players are actually from different schools and do not know each other, playing physical activities together seems to expedite ice-breaking and promote conversations.

5.3 Swan Boat

Swan Boat [1] is a multiplayer game designed to add fun to treadmill running which can be otherwise extremely boring. Two runners form a team, which simulates a two-seater pedalo, i.e. the pedal-powered recreational boat. A course is given to the teams. Team mates should tightly collaborate to maneuver their boat to clear the course faster. The basic means of propulsion and steering are their paces. A differential pace between them steers the boat to port or starboard. Matching their paces propels the boat straight. To implement Swan Boat, we used an interactive treadmill which automatically adjusts its speed to users' desired running speed [9]. Optionally, we also allowed players to use other smart exercise devices reporting exercise speed in real-time, such as jump rope, hula hoop, and stationary cycle, as described in [19]. (See Figure 4)

Items and gestures even more encourage dynamic interaction and team-competition. Various items with indigenous effects are drifting on the water. Gestures also enrich realistic game play. For example, a punch can be used to attack a nearby boat, hindering their play.

We installed four interactive treadmills for Swan Boat to study players' experiences and recruited 11 university students within the ages of 20 to 25 years old. We asked them to play Swan Boat for one week, and conducted an open-ended exit interview. The overall impressions were fairly positive and most of the participants strongly agreed that Swan Boat helped them to be better immersed in running. They elaborated that major fun factors were active participation and competition utilizing intuitive physical interactions.

6. CONCLUSION

We have presented our concept and initial prototype of Player Space Director (PSD), a mobile platform for pervasive games, targeting to fully reflect player's real-life situations. PSD provides an intuitive, declarative query language, CIQ, to specify diverse contextual interactions, facilitating the designers to develop pervasive games. PSD flexibly orchestrates game input devices in awareness of device availability and resource status. Given CIQs and device availability, PSD incorporates diverse processing modules for player space contexts, gestures, etc. PSD also provides in-situ or remote multi-player support with heterogeneous devices. We implemented a prototype system on mobile devices with a diverse set of sensors and actuators. PSD is applied in real pervasive game development, and those prototype games are experienced by lots of publicly recruited players.

In conclusion, PSD has well demonstrated an initiative toward the mobile pervasive game platform. The next step of PSD is to get out of the lab settings and immerse it into diverse real life situations.

7. REFERENCES

[1] Ahn, M., et al. Running or gaming, In Proc. ACE 2009, ACM.

[2] Ballagas, R. A., et al. REXplorer: a mobile, pervasive spell-casting game for tourists. Ext. Abstracts CHI 2007.

[3] Bao, L. and Intille, S.S. Activity recognition from user-annotated acceleration data. Pervasive 2004.

[4] Bjork, S., et al. Pirates! Using the physical world as a game board. Interact 2001.

[5] Chang, Y., et al. Playful toothbrush: ubicomp technology for teaching tooth brushing to kindergarten children. CHI 2008.

[6] Consolvo, et al. Activity sensing in the wild: a field trial of ubifit garden. CHI 2008

[7] Ferreira, P., et.al. A Middleware Architecture for Mobile and Pervasive Large Scale Augmented Reality Games. CNSR 2007.

[8] FFTW. http://www.fftw.org

[9] Frevola. http://www.r1solution.com

[10] Grønbæk, K., et.al. iGameFloor - a Platform for Co-Located Collaborative Games. ACE 2007.

[11] HUINS. http://www.huins.com

[12] Kang, S., et al., Seemon: scalable and energy-efficient context monitoring framework for sensor-rich mobile environments, MobiSys 2008.

[13] Lee, Y., et al., MobiCon: A Mobile Context-Monitoring Platform. *Communications of the ACM*, 55 (2012), 54—65

[14] Lin, J.J., et al. Fish'n'Steps: Encouraging Physical Activity with an Interactive Computing Game. UbiComp 2006.

[15] Linner, D., et al. Context-aware Multimedia Provisioning for Pervasive Games. IEEE ISM 2005.

[16] Magerkurth, C., et al. A Multimodal Interaction Framework for Pervasive Game Applications. AIMS 2003.

[17] Musolesi, M., et al. The Second Life of a Sensor: Integrating Real-world Experience in Virtual Worlds using Mobile Phones. EmNets 2008.

[18] Nenonen, V., et al. Using Heart Rate to Control an Interactive Game. CHI 2007.

[19] Park, T., et al. Exerlink – Enabling Pervasive Social Exergames with Heterogeneous Exercise Devices, MobiSys 2012.

[20] Ravindranath, L., et al. Code In The Air: Simplifying Sensing and Coordination Tasks on Smartphones, Hotmobile 2012.

[21] Wii Sports, http://www.nintendo.com/games/detail/1OTtO06SP7M52gi5m8pD6CnahbW8CzxE