From the Virtual to the Physical: The Gradual Transition to Pervasive Games

Carsten Magerkurth
Fraunhofer IPSI
AMBIENTE
Dolivostraße 15
64293 Darmstadt, Germany
magerkurth@ipsi.fhg.de

Carsten Röcker
Fraunhofer IPSI
AMBIENTE
Dolivostraße 15
64293 Darmstadt, Germany
roecker@ipsi.fhg.de

Timo Engelke
Fraunhofer IPSI
AMBIENTE
Dolivostraße 15
64293 Darmstadt, Germany
engelke@cs.brown.edu

ABSTRACT

In this paper, we present a component-based architecture for developing pervasive games that can flexibly integrate both graphical and tangible user interfaces. It allows for gradually augmenting purely virtual games with elements from the real world, thus transferring computer entertainment to our physical realities.

As a proof of concept, we have implemented a tabletop role playing game called Caves & Creatures that can be played with varying proportions of physical and graphical interface components and will provide a test bed for different interaction device ensembles.

1. INTRODUCTION

Nowadays, most computer games are controlled via graphical user interfaces with keyboards, joysticks or mice as the only interaction devices. Due to the tight coupling of these controllers with the graphical representations of the games on the screen, the interaction is usually focused on the display and disregards properties of the physical and social environments. Consequently, despite all technological drawbacks, other types of games such as traditional free form or board games have remained popular for thousands of years, because they obviously satisfy complementary needs such as the exertion of physical activity or social, face-to-face interaction. These properties are hard to realize via screen and keyboard alone (Mandryk & Inkpen, 2002).

To make computer games attractive to a larger target group that appreciates tangible game elements and interaction metaphors from outside the domain of computer games, novel physical gaming interfaces have emerged. Examples include the physical GameTrak boxing or baseball interfaces (http://www.in2games.uk.com), the Playstation Eye-Toy (http://www.eyetoy.com), or more recently the controllers of the Nintendo Wii video game consoles (http://wii.nintendo.com). The tremendous success e.g. of the Eye-Toy (Minkley, 2003), a camera add-on for the Sony Playstation that integrates the filmed shape of the player as an action object in the respective video games, is a clear indication that dedicated physical interfaces involving real world skills are an important future trend for entertainment applications.

Within computer game research, there is a complementary trend to augment traditional video games with aspects from the real world (e.g. Benford et al. 2005, Bjork et al. 2001, Cheok et al. 2002, Magerkurth et al. 2005) by integrating pervasive computing technologies. These hybrid or pervasive games combine the virtual nature of traditional video games with physical and social context, thus creating immersive gaming experiences that pervade the boundaries of virtual, physical and social domains. While many of these pervasive games focus on integrating entire buildings, streets, or even cities (e.g. Flinham et al. 2003) into a hybrid gaming experience, we address pervasive tabletop games that provide tangible interfaces borrowing interaction techniques from traditional board games. These pervasive board games attempt to bring together the advantageous elements of traditional tabletop games and computer entertainment technologies. The former emphasize the direct interaction between human players that face each other at an intimate distance. The latter provide rich audio and visual support, artificial intelligence, and several other features (cf. Mandryk & Inkpen 2002).

Figure 1: A Pervasive Tabletop Game

Figure 1 shows such a pervasive tabletop game, "Caves & Creatures", that is discussed in a later section of this paper.

2. DEVELOPMENT ISSUES

When it comes to the actual realization of pervasive games, developers quickly find themselves faced with new complexities in addition to the already complex process of creating traditional computer games. Creating pervasive games involves the integration of physical, social, and virtual aspects to be taken into account both at the game design and at the software engineering level. What might work well for traditional tabletop games (e.g. rolling dice for variability in the game flow) might work very differently in computer games (random number generation without manual intervention). Also, certain user interfaces that are designed for remote multiplayer usage in computer games (e.g. microphone input for voice commands) might totally break the social situation of co-located tabletop gaming.

Hence, for the development process of a pervasive game, it is crucial to be able to quickly prototype and tweak game mechanics as well as the involved user interfaces and interaction devices. One of the most important challenges for the development of pervasive games is thus the provision of a flexible infrastructure to dynamically integrate and replace various game components so that a test bed and environment for rapid prototyping can be realized.

In the following section, we outline our coordination and communication infrastructure called Pegasus that facilitates the development of pervasive games, before we afterwards discuss our sample implementation of a tabletop role playing game built upon Pegasus.

2.1 The Pegasus Infrastructure

The coordination and communication infrastructure 'Pegasus' provides the functionality to let heterogeneous software components connected in an ad-hoc manner share information and synchronize distributed data objects. It is designed as a lightweight communication solution which is capable of integrating also resource-constrained devices such as PDAs or custom interaction devices powered by diverse operating systems and providing wired or wireless communication protocols. The main design considerations are briefly outlined before the actual system architecture is presented.

Distribution and Dynamic Device Integration

Pegasus does not rely on a central server component, but allows communicating entities to directly exchange information. Physical interaction devices and their corresponding software proxies can be added and removed at any time with their state information being synchronized over an anonymous, decoupled communication bus. In the same way, the application logic can be distributed among communicating entities so that the disintegration of any single central server component does not necessarily have fatal consequences. Its highly dynamic nature regarding the integration and disintegration of communicating entities enables experimentation with different device setups during runtime.

Decoupled Communication

For the design of a distributed pervasive computing coordination infrastructure, the choice of an appropriate communication model is most crucial, because of its implications for scalability and flexibility. Pegasus consequently adheres to a publish/ subscribe interaction scheme in order to realize a loose coupling of participating components.

Subscription Scheme

In order to save communication resources in a publish/ subscribe system it is important to specify which particular events a communicating entity is interested in, so that subscribers do not have to receive all events that are published. Due to the small-scale nature of typical pervasive gaming applications that involve only a limited number of communicating

entities, the overhead of sophisticated content-based protocols is uncritical. Therefore, Pegasus follows a content-based approach that allows filtering events based on the evaluation of predicates that can be dynamically altered during runtime. Events are stored as distributed data trees / XML structures that are hierarchically ordered, so that the communication scheme of Pegasus can be seen as a combination of hierarchical and content-based systems. Accordingly, Pegasus introduces the notion of Functional Objects that get triggered on freely configurable changes within distributed data trees. A Functional Object subscribes to an arbitrary tree or sub-tree and is triggered on the evaluation of predicates that relate to changes in the subscribed tree. This is explained in more detail in the respective section of the system architecture.

2.1.1 System Architecture

The Pegasus system architecture comprises functionality on three layers of abstraction. These include 1.) the Basic Tools Layer that provides low level functions for dealing with data trees, network transfer, and XML parsing, 2.) the Network Data Layer that abstracts access to shared information via Accessor objects and 3.) the Functional Object Layer that provides various Functional Objects with multiple event handlers.

The functionalities on each of the layers are now discussed in more detail.

2.1.1.1 Basic Tools Layer

The Basic Tools Layer comprises various lightweight XML-related library functions in which a document class represents the data tree. There are several methods for accessing the data structures via paths. Furthermore, various network related functions for establishing connections between multiple Pegasus software components and transferring data between them are provided at this layer. With only the Basic Tools Layer available, we can already load a data tree out of an XML file, establish a connection and transmit parts of the data tree to another Pegasus instance via different communication methods such as a socket, a serial line, or a Bluetooth connection.

2.1.1.2 Network Data Layer

The second layer is the Network Data Layer. It is responsible for the abstraction of access to shared information across Pegasus instances. It defines several classes of objects that closely work together. The most important one is the Accessor object that is capable of holding a tree structure either loaded from a file or referenced from a part of a tree of another Accessor. On a change of the referenced data, e.g. from another Accessor instance, the Accessor receives a notification. Similarly, the Accessor can inform other objects of its own change. Accessors hence provide the capability of representing and synchronizing arbitrary data trees among distributed software components based on Pegasus.

Another important class at the Network Data Layer is the Gateway Accessor. It passes information from published Accessors to a corresponding Gateway inside a different Pegasus instance from which other Accessors can access the corresponding data. Using this mechanism, two or more instances can refer to the same data, even when they reside on different devices with different operating systems. As the Gateway wraps the entire functionality of cross process data transfer, adapters for various communication protocols can be added by deriving from the Gateway class. At the current point in time, connections via TCP/IP sockets, represented by a TCP Server- and a TCP Client Gateway are implemented as well as Gateways for Bluetooth and serial communications. Other connection types such as IrDA or HTTP are trivial to integrate by deriving from the Gateway class. By using Gateway Accessors, the idiosyncrasies of specific transport protocols are mitigated.

2.1.1.3 Functional Object Layer

The third layer of the architecture is the actual Functional Object Layer that implements the aforementioned Functional Objects. Functional Objects build upon the communication infrastructure of the Accessors requiring only code for the additional functionalities they provide. A Functional Object augments the methods for data access and synchronization that an Accessor provides. It can be informed by other Functional Objects or Accessors of data changes and can evaluate certain conditions in respect to these changes. This can result in calling a virtual Do()-method of the Functional Object class (which is the actual interface between custom code and the distributed application data) to which the respective data is passed as parameters. The object itself can change data, which in return can result in "calling" actions on other objects, simply by changing the data regarded by them.

Obviously, this Functional Object concept relates closely to the object oriented programming approach.

Every object has its own data structures and certain methods which can be called on different data changes. The class definition is a hierarchic structure that can be derived from. With this concept, we can create networks of Functional Objects that react on data changes appropriately for their defined situations.

2.1.2 Representation of User Interface Components

Each user interface component is represented by a software proxy that uses a Functional Object to synchronize with its respective data tree. The connection and graceful disconnection of components is handled by each component's Gateway Accessor. Since multiple Functional Objects can access a shared branch of data and anonymously inform each other of data changes, it is trivial to e.g. have a tangible game board with an associated data tree and then start up a graphical representation of the board that references the same data tree.

Data changes in one of the components would inform the other component of the respective changes and trigger appropriate actions. In this specific example, the Functional Object of the tangible game board would, by default, reverse any incoming data changes to enforce consistency with its own physical representation, whereas the graphical game board would accept any exterior data changes and update its graphical representation. Due to the decoupled communication scheme of Pegasus, an arbitrary amount of similar user interface components can be ad-hoc connected to a game application and automatically keep synchronized without any central coordination instance.

3. THE CAVES & CREATURES GAME

To demonstrate and explore the benefits of flexibly integrating various interaction devices to a rather complex pervasive gaming application, we have developed a "Dungeons & Dragons" style tabletop role laying game called "Caves & Creatures". We have chosen this type of game, because it offers the chance to implement it with varying degrees of pervasiveness.

The spectrum ranges from a traditional tabletop role playing game without any computer support at all to a purely virtual computer game that utilizes standard graphical user interfaces, mice and keyboards exclusively. In-between these extremes, it is possible to replace traditional or GUI-components, respectively, with novel user interfaces that retain the interaction metaphors known from the real world (e.g. shaking dice, using a magic wand), but that establish the link to the virtual domain by being unobtrusively augmented with sentient information technology.

3.1 Pervasive Computing User Interfaces

Examples for these interfaces include smart playing cards, game pieces, a dice cup, and a magic wand. For instance, the game integrates RFID augmented playing cards that represent items, weapons, armor, spells to be found, worn, used, cast, and traded between players. A physical game board with RFID augmented pieces can be used for positioning and moving game characters. A smart dice cup implements the rolling of dice and gestures performed with a magic wand determine the success of magic spells. The respective physical interfaces are briefly discussed in the following sections.

3.1.1 Smart Playing Cards

Playing Cards (see figure 2) are common interaction devices for many traditional games. In a computer augmented version, they retain their interaction properties. Due to RFID tags glued to the backs of the cards, the event of being played is detected by a stationary RFID reader.

Figure 2: Smart Playing Cards

While trading cards between players cannot be detected by a computer application with one stationary reader alone, the individual possession of cards can easily be reflected in the virtual world by utilizing multiple readers, possibly even integrated in the clothing of the players or in augmented bracelets (Smith et al. 2005).

3.1.2 Game Boards and Pieces

The interaction with physical game boards is a prototypical example for spatial tangible user interfaces (TUIS) that Ullmer & Ishii (2000) identify as the primary TUI approach in which artifacts are directly interpreted and augmented by a virtual application, not involving any additional layer of indirection.

Figure 3: Augmented Playing Pieces

Accordingly, a game board is a tangible interface that seamlessly integrates representations and controls and is thus preferable to graphical user interfaces in which spatial relationships are controlled in a different way (via the mouse) than they are represented. Figure 3 shows our tangible game board with RFID augmented playing pieces on it. Depending on the availability of such physical devices, the same game application might also utilize a virtual (GUI-) version of this type of interface.

3.1.3 The Smart Dice Cup

In order not to lose the physical and social aspects of rolling dice by simply creating random numbers in a computer application, we tried to preserve the multi-faceted nature of dice-rolling in our pervasive computing adaptation of rolling dice. Due to the size and feasibility problems associated with augmenting individual dice with respective sensor technology, we integrated multiple dice into one single smart artifact, the Smart Dice Cup (see fig. 4).

The augmentation of a dice cup allows for utilizing a physical manipulation technique (shaking) that influences the virtual outcome; a dice box is also a well-known interaction device for games that players are used to. Of course, a natural drawback of the approach is that a dice box is not identical to rolling physical dice and some players might not be used to using a dice box, although with games requiring multiple dice to be rolled simultaneously (such as Yahtzee or several role-playing or tabletop conflict games) it is common to use such a device, in fact, most editions of Yahtzee are shipped with dice boxes.

Figure 4: The Smart Dice Cup

The interaction was designed to be as similar to a traditional dice cup as possible. To generate random numbers, the device is lifted, shaken, put on a plain surface upside down, and then finally lifted again to see the results. However, in contrast to traditional dice, the sum of the spots is not counted from the physical dice after being tossed on the surface of the table. Instead, the spots are displayed via light emitting diodes (LEDs) on the surface of the dice cup top.

Shaking the device also emits a sound mimicking the sound of shaking a traditional dice box, although the integrated sound hardware does hardly deliver sound of acceptable quality. Since the smart dice box is capable of communicating with the environment via radio transmission, it is more preferable to let another sound source outside the device perform the respective audio output.

3.1.4 The Magic Wand

Finally, the magic wand is an interaction device that follows the approach of linking the exertion of a physical skill to a respective effect in a virtual application. It consists of a stick augmented with an accelerometer in its head that can be swung in a similar way as a conductor's baton or a magic wand (ref. Ciger et al. 2003). It picks up and digitally

converts the radial movements of the stick to discrete acceleration measures. There are three possible operating modes of the device, each relating to different usage scenarios and applications.

3.1.4.1 Gesture Recognition

The primary operating mode of the device is gesture recognition (see also figure 1). In a typical pervasive gaming application, the device translates and maps the real world qualities of the user's gestures to a virtual representation that has a certain effect in the game. The more accurate a player is able to perform a set of given gestures, the more successful is his outcome in the virtual world. Hence, a computer game wizard is no longer mighty due to some numbers stored in her character database, but because of physical skills acquired by real experiences.

3.1.4.2 Intensity Measurement

The gesture recognition mode works by matching the features of a set of stored gestures to the current incoming stream of data from the device. Another way of using the gesture based interaction device is by regarding the magnitude of the raw sensor data in order to measure the force of the swinging. By doing so, it is possible to create pervasive games that use the gesture based interaction device like a hammer or a sword instead of a wand or a conductor's baton.

3.1.4.3 Pointing

The final operating mode requires an RFID antenna built into the device's head. One can equip arbitrary artifacts with RFID tags that can be read and unambiguously identified by the antenna's head. This allows for unobtrusive multimodal interaction styles that follow Bolt's paradigm (Bolt 1980) by naturally specifying source and/ or target artifact of an arbitrary action.

For pervasive tabletop games that are in the domain of role playing games (such as the Caves & Creatures application), especially the capability of mapping the real-world skill of operating the physical device with the virtual effects of casting magic spells is an interesting feature that showcases the interaction between virtual and physical domains in pervasive games.

3.2 Game Mechanics

Using the interaction devices described above or their traditional virtual counterparts (from which some are shown in figure 5), it is possible to play the game with different degrees of pervasiveness. The actual game play of "Caves & Creatures" closely resembles that of the original "Dungeons & Dragons" tabletop miniatures game including also the more sophisticated rules for flanking or commander effects (cf. www.wizards.com).

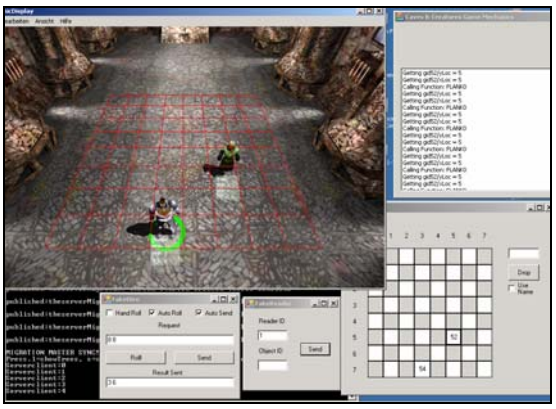

Figure 5: Some GUI Components of "Caves & Creatures"

The advantage of the game's architecture lies in the complete decoupling of UI components from the actual gaming applications. For instance, it is irrelevant, if a physical game board as in figure 3, a 3D rendered display of the board (figure 5, top left area), or a simple 2D GUI control (figure 5, bottom right area) is used to control the movements of the pieces. The latter two components can even be executed multiple times and keep synchronized automatically due to the anonymous Pegasus communication bus.

This makes it possible to play the game with only virtual UI components, with physical interaction devices, or with a combination of both. Likewise, it is technically indifferent, whether the game is played co-located like a traditional tabletop or board game, or whether the individual players (and their respective user interface components) are distributed and connected via the internet.

Table 1 recapitulates some of the game elements for which alternatives exist regarding the integration as a virtual or as a physical interface.

Table 1: Game Elements and their Interfaces

Game Element	Virtual Interface	Physical Interface
Moving and Selecting Playing Pieces	Graphical Game Board	RFID augmented Game Board
Readying Equipment, Weapons, & Armor	GUI application	RFID augmented playing cards
Generating Variability	Random number generator	Smart Dice Cup
Casting Spells	Random Number Generator	Magic Wand
Atmospheric Display	Background sound & music	Background sound & music, room illumination, ambient displays

The atmospheric display in the last row of table 1 is restricted to background sound and music in a virtual realization, whereas in a typical pervasive computing approach, the entire physical environment becomes part of the game. In Caves & Creatures, the illumination of the room (cf. the red glowing lamp behind the player on the right in figure 1) can be adapted to the current game situation as well as ambient information being shown on the wall displays in the room (see figure 1).

When we regard the multitude of possible game realizations that involve different virtual and physical interfaces and co-located and remote player distributions, it becomes clear that the evaluation of such a system is a complex task. So far, we have laid the foundation for rapidly experimenting with different setups. As a next step, we will conduct a user study to find out which degree of pervasiveness is most appreciated by the players and which game elements should be realized in a specific way. Ultimately, we hope to gain insights on the question, if the diverse game genres of video games and traditional tabletop games can really be united with pervasive games, or if it finally turns out that most players prefer either the traditional non-computer realization or the purely virtual video game, but do not make the transgression to pervasive games.

4. CONCLUSIONS

Pegasus, the component based architecture for pervasive gaming applications allows for developing pervasive games without anticipating exact configurations of interaction devices. This reduces the complexity of game development and opens up the chance of experimenting with different configurations.

As discussed in this paper, the sample game “Caves & Creatures” can be played without any physical interface components as well as with the specialized devices we have developed such as the magic wand or the Smart Dice Cup. This allows for systematic evaluations of appropriate interface compositions and game designs that we will conduct in the future.

5. RELATED WORK

There are several projects in the computer gaming research community that address the integration of the real and the virtual world. For instance, the academic research project False Prophets (Mandryk & Inkpen 2002) is a pervasive board game, in which players jointly explore a landscape on a physical game board. A custom crafted infrared sensor interface helps identifying the playing pieces, while the game board is projected on the table. The realization of False Prophets is similar to parts of our platform, even though it is currently limited to a single exploration game. In the same spirit, but technically more constrained due to its very early realization in the mid-nineties is also the Digital Playing Desk from Rauterberg et al. (1996).

Bjork et al. (2001) presented a hybrid game system called Pirates! that adds the world around us to gaming applications with players moving in the physical domain and experiencing location dependent mini-games on mobile computers. Thereby, Pirates! follows a very interesting approach to integrate virtual and physical components in game applications. Unfortunately, the mini-games on the PDAs do not involve multiple players, so that social aspects are not very relevant for Pirates!

From the domain of augmented reality are the works of Cheok et al. (2002). These augmented reality approaches involve the use of cameras and specialized AR glasses that project digital information over the standard camera images. The results create visually stunning hybrid words that involve tangible and artifacts and digital augmentations. The drawback, however, is that AR glasses need to be worn that might hamper social interaction, because players lose direct eye contact. Flintham et al. (2003) present a

large scale game played on the real streets that also integrates players connected via the internet hence also pervading multiple realities.

Details on several other pervasive gaming projects can also be gained from a recent overview article (Magerkurth et al. 2005).

6. ACKNOWLEDGMENTS

We thank all our colleagues and friends at Fraunhofer IPSI for revising earlier versions of this paper.

7. REFERENCES

Benford, S., Magerkurth, C., Ljungstrand, P. (2005) Bridging the Physical and Digital in Pervasive Gaming. In Communications of the ACM, March 2005, vol. 48. No. 3. Pages 54-57.

Bolt, R. A. (1980). “Put-that-there”: Voice and gesture at the graphics interface. In Proceedings of SIGGRAPH '80. ACM Press, New York, NY, 262-270.

Bjork, S., Falk, J., Hansson, R., and Ljungstrand, P. Pirates! using the physical world as a game board. In Proceedings of Interact 2001, 2001.

Cheok, A. D., Yang, X., Ying, Z. Z., Billinghurst, M., Kato, H. (2002) Touch-Space: Mixed Reality Game Space Based on Ubiquitous, Tangible, and Social Computing. In Personal and Ubiquitous Computing (2002), 6: 430-442.

Ciger, J., Gutierrez, M., Vexo, F., Thalmann, D. (2003). The magic wand. 19th Spring Conference on Computer Graphics (Budmerice, Slovakia, April 24 - 26, 2003). L. Szirmay-Kalos, Ed. SCCG '03. ACM Press, New York, NY, 119-124.

Flintham, M., Anastasi, R., Benford, S., Drozd, A., Mathrick, J., Rowland, D., Oldroyd, A., Sutton, J., Tandavanitj, N., Adams, M., Row-Farr, J. (2003) Uncle Roy All Around You. Level Up conference, Utrecht, Nederlands, 2003.

Floerkemeier, C., Mattern, F. (2006). Smart Playing Cards – Enhancing the Gaming Experience with RFID, www.pergames.de

Mandryk, R., Inkpen, K. False prophets: Exploring hybrid board/video games. In Extended Proceedings of CHI 2002, pages 640–641. ACM Press, 2002.

Magerkurth, C., Cheok, A.D., Mandryk, R.L., Nilsen, T. (2005): Pervasive Games. In: ACM Computers in Entertainment, Vol. 3, No. 3, July 2005.

Magerkurth, C., Engelke, T., Memisoglu, M. (2004) Augmenting the Virtual Domain with Physical and Social Elements. In ACM Computers in Entertainment, Vol. 2, No. 4, October 2004.

Minkley, J. (2003). Eyetoy shifts a million. Online article at Computer and Video Games Market. http://www.computerandvideogames.com/news/news_story.php(que)id=97876

Rauterberg, M., Mauch, T., Stebler, R. (1996). The Digital Playing Desk: a Case Study for Augmented Reality. 5th IEEE Workshop on Robot and Human Communication, Tsukuba, Japan, 410-415.

Smith, J. R., Fishkin, K. P., Jiang, B., Mamishev, A., Philipose, M., Rea, A. D., Roy, S., and Sundara-Rajan, K. 2005. RFID-based techniques for human-activity detection. Commun. ACM 48, 9 (Sep. 2005), 39-44.

Ullmer, B., Ishii, H. (2000). Emerging frameworks for tangible user interfaces. IBM Systems Journal, 39(3):915–931.