Designing Enhanced Authoring Tools for Pervasive Games

Alf Inge Wang
Norwegian University of Science and Technology
alfw@idi.ntnu.no

Audrius Jurgelionis*
Norwegian University of Science and Technology / Fraunhofer FIT
ajurge@gmail.com

Hong Guo and Hallvard Trætteberg
Norwegian University of Science and Technology
{guohong,hal}@idi.ntnu.no

ABSTRACT

This paper describes a general gameplay model and authoring tool for construction of pervasive games. The proposed gameplay model is relying on the definition of common constructs among a range of pervasive game categories that are broken down into atomic game elements that are later used to build various games. Further, the paper presents design of a platform independent authoring tool, focusing on the GUI and its easiness to use, and envisions a possible implementation for the described gameplay model. The authoring tool targeting non-technical users and is intended for integration with other software tools and underlying platforms to build a powerful framework for pervasive game development.

Categories and Subject Descriptors

H.5 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*Artificial, augmented, and virtual realities*; D.2.11 [**Software Engineering**]: Software Architectures—*Data abstraction, Domain-specific architectures*

General Terms

Design, Theory

Keywords

Pervasive games, authoring tool, gameplay model

1. INTRODUCTION

Modern pervasive games encompass multiple dimensions of physical and virtual worlds. Usually the four dimensions covering most of the diverse pervasive game world are perceived as spatial, temporal, perceptual and social perspectives [1]. These provide an enormous pool of possibilities for game designers who can create and design an unlimited number of interactive pervasive games. However, when it comes to implementing such pervasive games, innovative ideas and designs often bounce back from the obstacles raised by the gap between the technical implementation needs and the capabilities of today's enabling technologies.

In the past few years there have been numerous attempts to provide game designers with tools that would ease the development of a variety of pervasive games. Unfortunately, the diversity of pervasive games and the constant change of technologies prohibit a fast growth of such frameworks. The most prominent discontinued examples of such frameworks were presented in [2], [3], [4] and [5]. These frameworks fairly accurately defined the functional and operational requirements within the chosen game domain. E.g. the authors of [2] focus on "capture the flag" games. Their main goal was to provide a domain oriented model that supports the game designer with suitable abstractions in order to fulfill the functional requirements of a specific game. In addition, the framework in [2] aims to provide an abstraction of the technical infrastructure for the game implementation. Thus, offering to the game designer a mean to focus on the logic of the game instead of coping with technical issues of the underlying platform. However, considering their application to a wider pervasive domain, most of the frameworks fall short. It is conditioned by the fact that almost all requirements for these frameworks could be traced back to the designed games. This makes their application for other pervasive game domains very limited. Importantly, the mentioned frameworks still require the game designer to have a substantial knowledge in programming. This is a serious limitation for game designers with little or no programming skills, which is the target audience for our approach.

The need for an easy to use content creation interface has been addressed in numerous publications [6], [7], [8] and [9]. They are mostly focusing on interfaces for interactive story creation and authoring. The main goals have been to reduce the effort and enabling non-programmers to create elaborated contents [6]. Similarly, the authors of [9] introduced an authoring tool for the Backseat Playground pervasive game. The mentioned authoring tools are limited to their application domain, which is very narrow, since all of them are suited only for a particular type of content. Although the idea of having such an authoring interface is very welcome in the pervasive game domain as a whole, it would be difficult to adapt such authoring tools to pervasive game creation. These interactive story authoring interfaces are still relatively complex to use. Thus, considering the

*This work was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme.

fact that pervasive games require a number of components with different user interaction type and that the gameplay story could be very different and require access to different sensor data, it would be relatively difficult to extend them for pervasive game authoring. This problem was somehow solved by the TOTEM framework [10], which provides a very easy to use interface for mobile-mixed reality game authoring. Nevertheless, the possibility of creating different games is also very limited in this framework. In this paper, we suggest to extend the work on the TOTEM framework by adding a simplified but effective gameplay logic model that covers the basic features required to build pervasive games across different domains resulting in a framework that is very easy and flexible to use. Finally, another important remark is that most of the frameworks covered in this paper were designed targeting a single platform. Lack for cross-platform support in the framework design limits their application domain in long term, thus, this issue must be addressed as well.

This paper proposes a gameplay model and authoring tool that adopt the functional and operational requirements from the previously discussed frameworks and enhance them to cover a wider range of pervasive games. The paper is organized as follows: Section 2 presents the atomic gameplay model which is based on the TeMPS conceptual framework and other relevant game classification work, Section 3 describes and visualizes the proposed pervasive game authoring tool and its user interface, and Section 4 concludes the paper.

2. A PERVASIVE GAMEPLAY MODEL

This section describes a concept in which the analysis of several pervasive games has resulted in identifying atomic game elements, as the smallest playable parts of the game, that are later used to construct different games. The TeMPS conceptual framework defined in [1] is used as a point of departure for the general gameplay model. TeMPS systematically surveys and categorizes important aspects of over thirty pervasive and social games covering the four different perspectives temporality, mobility, perceptibility and sociality, as well as relevant technical implementation details. Typical gameplay elements can be to solve puzzles, discover and find locations and traces, combine information, find virtual and physical objects, social interaction with virtual or real players, move between locations more efficiently, compete in battles including both physical and virtual elements, trap virtual objects on physical locations, etc. This knowledge enables creating a general gameplay model, which covers a reduced set of components that are present in a range of different pervasive games, without actually sticking to a specific pervasive game or its genre.

Fig 1 shows our pervasive gameplay model with its main components and their relationships. In this model, gameplay is represented in a hierarchy of challenges which consist of some actions that are further connected to user inputs. Note that the list of actions and challenges in the model is not complete and new items can be added to the model as needed. The model distinguishes between physical actions the user performs, represented as actions, and their virtual representation in the game, represented as challenges. The action nodes connect our gameplay model with the TeMPS

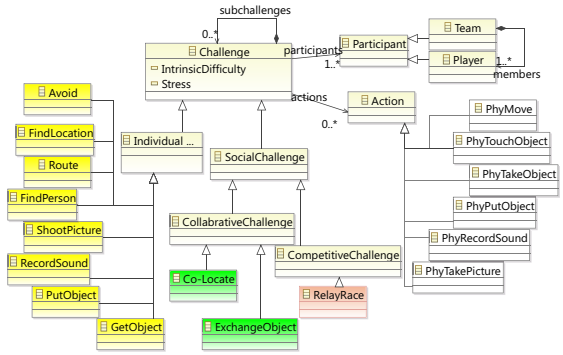

Figure 1: A Simplified Pervasive Gameplay Model

model on the perceptibility perspective. The perceptibility perspective in the TeMPS model enumerates various ways about how the game utilizes data from the physical world, or how the game influences the physical world, while the action nodes in our gameplay model are used to represent typical actions that the game requires, which usually correspond to some physical inputs to the game. For instance, the *'PhyMove'* action means a physical location information as an input to the game, and the *'PhyTakePicture'*, *'PhyTakeSound'*, *'PhyGetObject'* corresponds other aspects like graphics, video, sound as the game input. In order to cover the *social* perspective, the model must contain mechanisms to express how players interact. This is achieved by defining two different interaction blocks, competitive and collaborative. The competition block represents a single gameplay element or challenge (e.g. co-locate object or place) assigned to all players/teams. It specifies that the assigned gameplay element can be won according to a particular parameter, which can be points earned, time and/or location, e.g. a player that completes the challenge first. The collaboration block models how players or teams need to coordinate or collaborate their challenges in order to succeed, e.g. all players must have completed their tasks before any player can proceed in the game. One game can contain a mixture of individual, competitive and collaborative blocks to vary how players interact within one specific game. This means that team competition (in-team collaboration, inter-team competition) is also possible. The model shown in Fig 1 is used as input when designing and implementing the authoring tool described in the following section. The model identifies the elements to be included in the tool as well as how the different parts are related.

3. AUTHORING TOOL DESIGN

The game authoring tool should enable users to create and author pervasive games using a predefined set of gameplay elements. More importantly, it should offer an easy way of pre-configuring the required game elements and programming the gameplay rules and events. This includes the game interface design, media content creation/acquisition, access to underlying technologies and platforms, such as GPS, camera or Bluetooth, and their combination into a set of programmable atomic gameplay elements which will produce a game according to some user-programmed game rules. Since one of the identified drawbacks in the existing frameworks is related to their easiness of use by non-programmers, Google

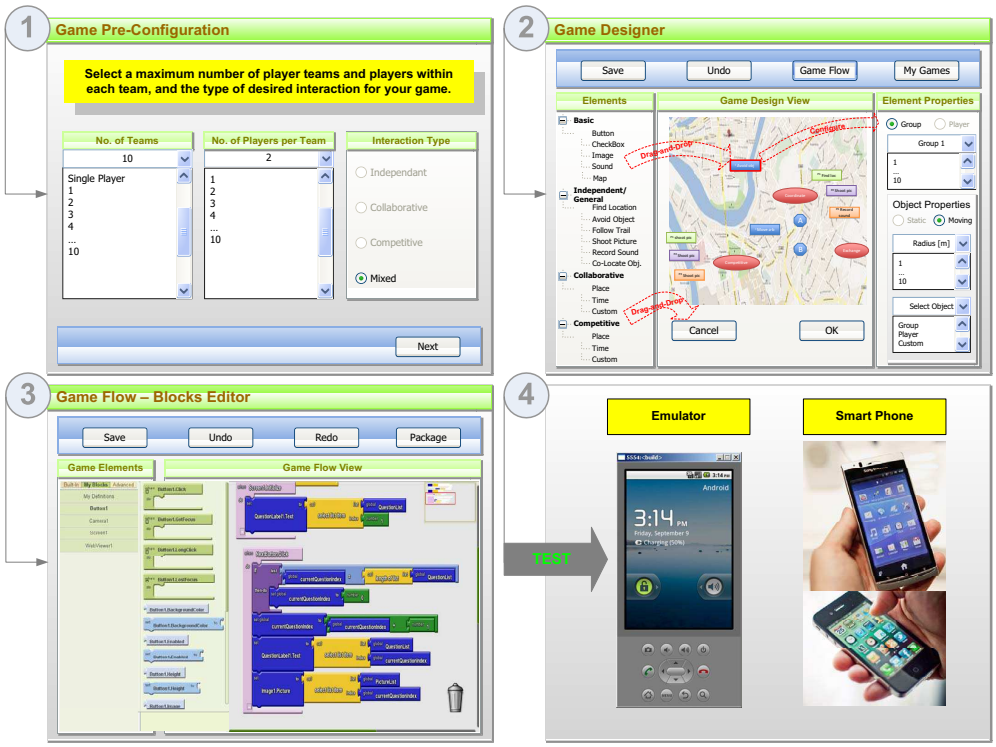

Figure 2: Pervasive Game Authoring Tool User Interface and Basic Design Flow

App Inventor application [11] is taken as a reference for the authoring tool design. With a strong focus on providing a relatively easy way to build applications for the Android platform, it is one of the best examples of user friendly and easy to use development environments. In addition, some studies conducted with users having little or almost no programming knowledge showed that similar tools enable them to create applications in an easy and quick way [12]. It uses a graphical interface, very similar to Scratch and the StarLogo TNG user interface. The application itself is based on the OpenBlocks library [12], which will be used in the proposed authoring tool design. OpenBlocks enables application developers to build and iterate their own graphical block programming systems by specifying a single XML file, which provides to application developers a lot of flexibility in selecting a programming environment and platform for the development of the software. On the other hand, it is needed to build a compiler/interpreter for the programming language created with the blocks editor. This of course requires some effort and time, but again enables a greater flexibility. In fact, the aim of the proposed authoring tool is to support multiple platforms. Thus, we have specified a general design of the user interface and pervasive game design flow, see Fig 2, without adhering to any platform or programming language.

The gameplay model defined in the previous section is used as a core element for the authoring tool design. All the gameplay elements defined in Fig 1 are instantiated by the authoring tool and presented in the graphical user interface (GUI) where some of their properties can be configured. Process of creating a new pervasive game consists of four main steps. In the *first* step, the designer is required to narrow down the palette of game elements, depending on the selected game interaction type, and define a number of possible game players/groups per game. During this step, a virtual backpack container must be created and defined. The virtual backpack defines all the information that will be related to the player during the game, and thus define the player state that can change over time. A virtual backpack can typically contain variables to represent points, collected items, achievements, lives, completed gameplay elements, etc. The virtual backpack is typically used to administrate player progress, coordinate players, evaluate winners in competitive challenges, etc. The *second step* includes designing the game interface, as it will seen by the end user, by configuring and placing various gameplay elements by placing different components in a 2D map. These components correspond to the ones defined in the general play model in Section 2. However, their relationships are still undefined at this point of the process. The GUI of the authoring tool enables assignment of the selected game element for each player or a team of players separately or as a whole (to enable team as well as mixed challenges). Additionally, it provides an interface for setting some static parameters for each game component. Note that the gameplay elements represent pre-implemented parts of the gameplay that can be combined into one game, including functionality for location-awareness, communication, GUI-presentation and more. If the designer wants to add collaborative gameplay elements, she or he first has to place a collaborative meta-element on the 2D map that is used to configure the collaboration. Then the designer has to specify which tasks should be involved in the collaboration. The same applies for the

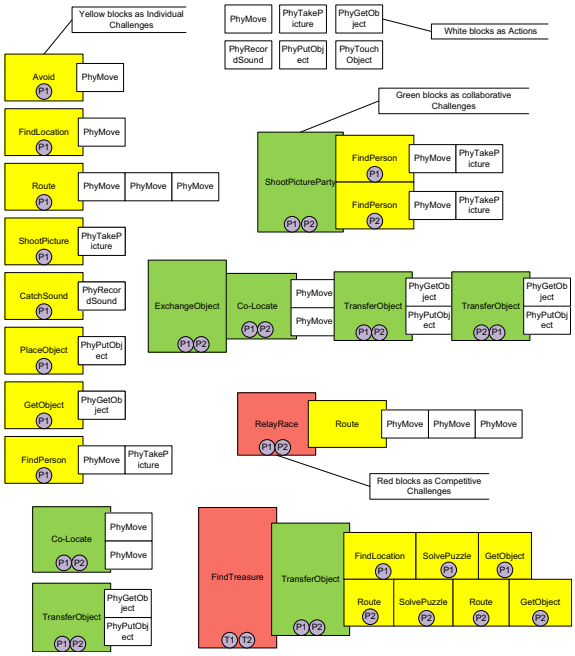

Figure 3: Visual Represantation of the Game Model Elements for Block Editing

competitive elements, where the designer can specify how the game user will win, e.g. highest score or within shortest time. The *third step* lets the designer to program the flow of the game utilizing the relationships between the selected game elements defined in Fig 1. The programming is a visual manipulation of graphical objects carried out by using an interface based on the OpenBlocks library, see Fig 3. This approach resembles Lego construction toys in which Lego bricks can be assembled and connected in many ways. In this step, it is demonstrated how the game designer could easily assemble basic and complex gameplay blocks together as a puzzle. Different colors are used to distinguish individual, competitive and collaborative challenges. The circles in the challenge blocks specify the players or the teams that the challenges are assigned to. The elements used in step three include gameplay elements from step two, the content of the virtual backpack (variables), the time and the location of the players. The *fourth step* includes compiling, packaging and testing of the developed game, which requires building a specialized compiler and packager for selected platform. The framework should allow testing both on an emulator and on the device.

4. CONCLUSIONS AND FUTURE WORK

This paper describes an initial proposal of a framework for allowing non-technical game designers to create a multitude of pervasive games using a GUI-based tool. The foundation of this approach is the atomic gameplay element model that enables combining gameplay elements extracted from various pervasive games in order to produce new games. Theoretically, it should cover most types of pervasive games. In addition, new pervasive gameplay elements can be added to the framework when needed, making it possible to evolve in the future, as technology and more pervasive games are available. The goal with our approach is to produce a tool that makes it possible for even teachers and students in school to create their own educational pervasive games that will enable learning that combines the virtual and the real world.

This paper only describes high-level ideas including the underlying model, the development process and the authoring tool. Future work will go into the details of the gameplay model, the semantics and how everything will be implemented. One of the major challenges with this approach is to make the implementation robust in such a way that the designer does not have to be concerned with the complex underlying technologies.

5. REFERENCES

[1] Hong Guo, Hallvard Traetteberg, Alf Inge Wang, and Meng Zhu. Temps: A conceptual framework for pervasive and social games. In *Proceedings of the 2010 Third IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning*, DIGITEL '10, pages 31–37, Washington, DC, USA, 2010. IEEE Computer Society.

[2] Jan-Peter Tutzschke and Olaf Zukunft. Frap: a framework for pervasive games. In *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, EICS '09, pages 133–142, New York, NY, USA, 2009. ACM.

[3] José Braz, Pere-Pau Vázquez, and João Madeiras Pereira, editors. *GRAPP 2007, Proceedings of the Second International Conference on Computer Graphics Theory and Applications, Barcelona, Spain, March 8-11, Volume AS/IE*. INSTICC - Institute for Systems and Technologies of Information, Control and Communication, 2007.

[4] D. Linner, F. Kirsch, I. Radusch, and S. Steglich. Context-aware multimedia provisioning for pervasive games. In *Multimedia, Seventh IEEE International Symposium on*, page 9 pp., dec. 2005.

[5] Carsten Magerkurth, Richard Stenzel, Norbert Streitz, and Erich Neuhold. A multimodal interaction framework for pervasive game applications. In *Artificial Intelligence in Mobile System (AIMS), Fraunhofer IPSI*, pages 1–8, 2003.

[6] Sebastian Sauer, Kerstin Osswald, Xavier Wielemans, and Matthias Stifter. U-create: Creative authoring tools for edutainment applications. In *Technologies for Interactive Digital Storytelling and Entertainment*, volume 4326 of *Lecture Notes in Computer Science*, pages 163–168. Springer Berlin / Heidelberg, 2006.

[7] Yun-Gyung Cheong, Yeo-Jin Kim, Wook-Hee Min, Eok-Soo Shim, and Jin-Young Kim. Prism: A framework for authoring interactive narratives. In *Interactive Storytelling*, volume 5334 of *Lecture Notes in Computer Science*, pages 297–308. Springer Berlin / Heidelberg, 2008.

[8] B. Medler and B. Magerko. Scribe: A general tool for authoring interactive drama. In *3rd International Conference on Technologies for Interactive Digital Storytelling and Entertainment*, TIDSE 2006, pages 139–150 pp., Darmstadt, Germany, December 2006.

[9] A. Bäckström and E. Danell. Authoring tools for interactive narratives - an interface design of a script editor for the pervasive game backseat playground. Master's thesis, Umeå University, Department of Computing Science, March 2009.

[10] TOTEM. Totem project overview. http://www.totem-games.org/?q=overview, September 2011.

[11] App Inventor for Android. About - app inventor for android. http://www.appinventorbeta.com/about/, September 2011.

[12] Ricarose Vallarta Roque. Openblocks : an extendable framework for graphical block programming systems. Master's thesis, Massachusetts Institute of Technology., May 2007.