International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 4 (2012) pp. 66–75

Gamework – A Framework Approach for Customizable Pervasive Applications

Christoph Stach

University of Stuttgart,
Institute of Parallel and Distributed Systems,
Applications of Parallel and Distributed Systems,
Universitätsstraße 38, 70569 Stuttgart, Germany
christoph.stach@ipvs.uni-stuttgart.de
https://www.ipvs.uni-stuttgart.de/de/institut/team/Stach/

Abstract: **The number of pervasive games is growing. In a pervasive game players have to interact with their environment in order to control their avatar. While in the past such a game required immense hardware equipment, nowadays, Smartphones have all required functionality built-in already. In addition, there is an upcoming trend towards software, supported with new content and knowledge by an active community. By combining these two trends, a new genre for computer games arises, targeting not only established gamers but also a new audience. In this paper we present a framework for pervasive games that support various customization techniques. Therefore, we divided the techniques into four classes depending on the player's technical knowledge and scale of adaption potential. Further, we present two customizable pervasive games we have implemented using the framework. Concluding, we show the potential of these games as well as the potential of our framework. We also report on early experiences in exploiting the customization approach of our framework.**

Keywords: Mobile Services, Pervasive Multi-player Games, Customizable Framework, Data Analysis and Improvement.

I. Introduction

With the introduction of the first context-aware devices, such as the so called active badges [1], Mark Weiser's visionary idea of powerful computer systems, which integrate seamlessly into our daily routine [2], started to become reality. Without question, these devices were accessible by a limited group of researchers, only and solely served the intended purpose – locating individuals within a building and managing access control. Since these early days of mobile context-aware – or pervasive – systems, technology made huge improvements. Alongside with an improvement in mobile devices' performance, the number of different sensors, which are included already, increases steadily. In addition to well-known components, such as sensors to determine the position of a user, we find intensity sensors, thermometers and even sensors for measuring medical data [3] in today's mobile devices. In fact, even conventional modern Smartphones possess not only powerful processors and plenty of main memory, but are also equipped with components such as a sufficient touch-screen, GPS sensors, Bluetooth, Wi-Fi, and much more. Since, on the one hand these devices are accessible to everyone and on the other hand we are having them always with us, they completely full-fill Weiser's future vision. Due to an increase in sales of modern Smartphones, the number of applications for these devices is growing rapidly. Especially the games sector registers high sticking cash out [4]. Although the number of pervasive games available on Smartphones is still relatively low, it makes sense to worry about reusable components and a reuse-oriented development methodology, nevertheless, in order to prevent inventing the wheel over and over again.

While pervasive games may differ in many aspects, they all map an extract of real world's context on a virtual game somehow [5]. Generally this happens through a GPS-based sensor which tracks a player's position and uses it to control the avatar in the game. Therefore, most common pervasive games use massive and expensive hardware assignments [6]. Albeit, there are approaches such as *AR-Soccer* [7], a penalty game with a virtual ball and goal, or *Capture the Flag* [8], where two teams have to defend a flag while capturing the opponent's one, almost all well-known pervasive games focus on augmented reality systems – computer-based systems stimulating any human sense especially vision [9] – including backpacked laptops or even head-mounted displays [10]. In *Human Pacman*, a pervasive version of the original *Pacman* game, a player controls either the *Pacman* or a ghost by moving through the streets of Singapore and has to collect points or catch the *Pacman* in order to win. Therefore, every player needs a special vest or a backpack equipped with nearly all components of a laptop [11]. The *REXplorer*, an educational city-game provided by the Regensburg Experience museum, does not need extensive technical equipment. Although, it still has to provide all players with special devices due to support gesture recognition in the game. Thus, the number of simultaneous players is limits heavily [12]. Even though, these games are a lot of fun, the effort needed to set up a game session is too high to arrange them regularly. Furthermore, the high hardware requirements make it practically impossible for a regular person to realize her own game ideas.

Therefore, we were endeavored to bring these two trends together: a framework supporting the development process of pervasive games for Smartphones [13]. By this means,

everybody is able to create her own game – at least with enough programming experience. It showed up that this was still a hurdle too high for most of the people. Hence, we had to find other ways which allow players to bring in their own ideas and alter or enrich the original game idea. Modern operating systems for mobiles enable everybody to develop their own additional applications for those devices [14]. Without such a customization feature, especially casual games might lose their attraction very rapidly [15]. In particular, pervasive games qualify for small customizations, as placing the playing field to another (real world) location, adding new tasks up to extending the game logic [16].

For this purpose, we initially identified different requirements towards such a framework e.g. various customization techniques. Afterwards, we analyzed what kind of game adaption each of them could be used for, classified our audience depending on their knowledge in programming and determined how each of them could be assisted by a framework. Based on this preliminary work, we implemented *Gamework*, a framework for pervasive games running on mobile devices in order to enable a shortened developmental period and to involve all players in the development process by providing the tools to customize a game to a certain degree. While some frameworks for pervasive games already exist (e.g. *FRAP* [17], *TeMPS* [18] or our former approach for mobile browser games [19]), our new approach supports customizable pervasive games which is a unique feature. The degree a game can be customized with our approach is determined by the game's nature as well as the player's expertise. But, even players with limited programming experience will be able to make small changes on existing games, while programming experts are supported in developing new games or even new game genres. To demonstrate this, we present prototypes of two pervasive games implemented with *Gamework*.

In the following we will discuss different requirements towards *Gamework* in Section II. Section III will describe the model of *Gamework* while Section IV will present two game prototypes implemented with it in more detail. Finally, Section V introduces additions to and extensions for *Gamework* before we conclude this paper with Section VI.

II. Requirements

After substantial analysis on current pervasive games, we realized that *Gamework* will have to fulfill three major requirements in order to enable a wide and successful spread of the games created with its help: Customization, data management and platform independence. We will look at these requirements in more detail in the following subsections.

A. Customization

In some preliminary work, we studied game's nature and how players can influence it [20]. As a result of this work, we were able to distinguish four different classes of customization techniques in games. These classes vary in effort as well as scope of supply and services. In the following, we describe how we support game customization by using context data or user-generated content in games and adapting the game-flow or even creating new games by only reusing the *Gamework*'s artifacts. We will conclude this subsection by a short comparison the four customization classes. As most of the current pervasive games do not support NPCs (non-player characters), we can ignore dynamic (AI) adaption techniques, as presented in Hunicke [21].

1) Context in Games

The easiest way to customize a game is setting the game-field to the current location of the player. This adaption technique does not require any programming skills from a player. Nevertheless, it will give a player an individual game experience because no other plays exactly that game. On the other hand, the game designer will have to include this customization ability in her game logic and not every type of game is appropriate handling with dynamic game-field positions. Because of the need to interact heavily with the environment, the real world setting of pervasive games should be chosen wisely, as natural obstacles can hamper accomplishing an aim unnecessary or even inhibit it completely. Christoph Schlieder et al. reckon that especially location-based versions of classical board games suits to be modified this way [22].

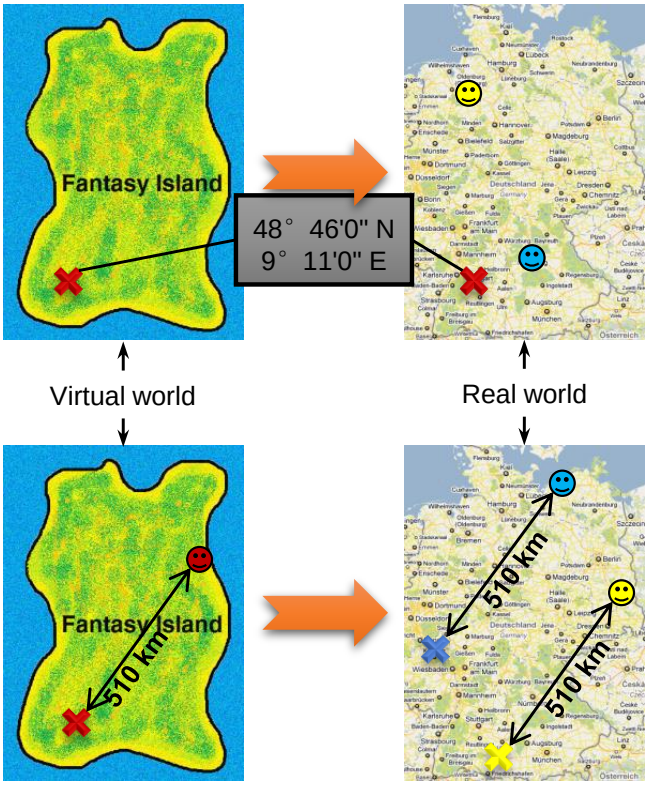

Figure. 1: Absolute (top) vs. relative (bottom) positioning

SYGo [23], which is actually a pervasive adoption of the famous German board game *Scotland Yard* by Ravensburger™, represents such a kind of game. While the board version of the game always takes place in the center of London, the game-field in *SYGo* is set relative to the player's current position. The player moves her avatar to field X by going to the corresponding location in reality. We call this method to use location in an application *absolute positioning*. In contrast, we speak of *relative positioning*, when the player's absolute position is ignored completely but her movement relative to

her starting position is used to steer an avatar. Thereby, new challenges occur such as what to do with unreachable regions because of obstacles or how to realize virtual barriers. Figure 1 shows the difference between absolute and relative positioning: In a game with absolute positioning an object (X) remains at a fixed location, no matter where a player enters the game. In contrast, a game using relative positions defines an entry point (red smiley) and determines the distances of all game objects to this entry point. According to these specified distances and the starting point of a player, all objects are set in the real world.
As *SYGo* uses absolute locations, all players are forced to gather within a certain region close to its center. We will even present a game without this restriction in Subsection IV-B.
As a matter of course, this kind of customization has to be provided by each game itself and not by a framework. Nevertheless, we tried to support the players who want to use such a feature in their games in the game development process by adding geographical functions to *Gamework*, e.g. for calculating the relative distance to a given location.

2) User-generated Content

Another way allowing players to customize a game is to give them the possibility to add new content to an existing game. While this is nothing new in the Web 2.0 environment, it's rarely used in games apart from a few exceptions such as Second Life [24]. We are convinced that especially location based games are suited for this kind of customization. John Krumm et al. [25] define pervasive user-generated content as the possibility to add some new information to a community-based system, e.g. some ratings to given points of interest. Figure 2 shows the idea of different groups using various techniques to generate any new content and add it to the game. Then, this new content is directly used in the game. Therefore, it requires many voluntary contributors. On the other hand the binding within the community is tightened.

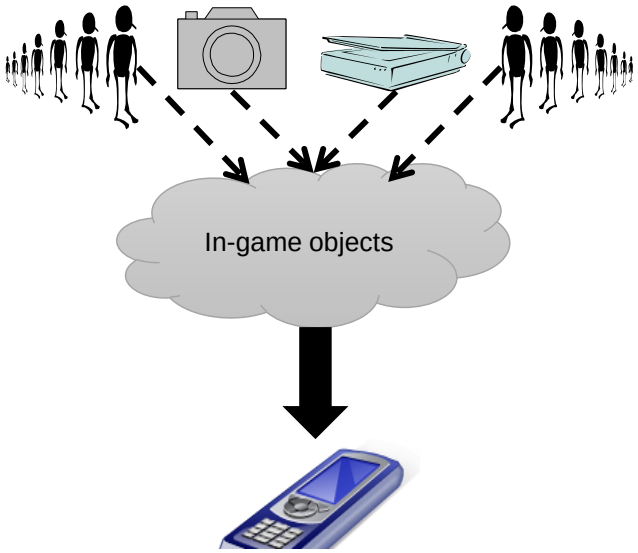

Figure. 2: User-generated content

E.g., user-generated content could be used to add a new target which actually implies creating a new level or to ease an existing search by adding some hints in a geocaching-like game. *Gamework* provides functions to store additional information such as pictures or some text to existing objects and allows the player to create completely new points of interest.
In addition, *Gamework* includes a map-editor with a GUI which allows even a non-technical player to create, modify or delete any object in the game just by clicking on a map. Nevertheless, this feature must be supported by the game. If a game manages its objects by itself, the *Gamework* editor cannot interact with them.

3) Game-flow Adaption

In the following we understand "game-flow" in the sense of application flow in the area of games and not in the sense of Penelope Sweetser and Peta Wyeth [26]. To customize games in a major way, its game-flow has to be modified. Mark John Taylor et al. [27] introduce a technique to visualize a flow-graph for every level in an UML-like style. In the graph, not only the game-flow with all alternatives to solve a level is represented but also the storyline and any scripted event. By merging all flow-graphs a single graph is generated which represents the whole game. To change the game-glow in order to create a new game all there is left to do is to modify or rearrange the graph's nodes or edges.
Figure 3 shows on the left side an example for a simplified finite-state machine (FSM) modeling the level-flow of one concrete game. It has to be read that when a player has entered level one she has to find the exit (1) in order to continue to the next level. If she even finds a shortcut (2), she can skip the next level and proceed to level 3. If she fails in this attempt (X), she has to leave the game. When we take a closer look into one certain level, all actions available in a certain situation can be modeled in the same way. The right side illustrates this by the example of level 3. When a player enters the first room she may either proceed to the next room unharmed, by finding the right key (1), or has to face a monster when she runs into it (2).
The easiest customization using this technique would be arranging the levels in a new order or removing some of them by moving or deleting some edges. Provided with some examples, an interested player should also be able to add further game states or transitions to the graph, after a while. We considered that technique already in the design phase of *Gamework* by representing every game as a finite state machine. So, a player who understands Java code is able to modify the game-flow, in a simple way.

4) Creation of New Games from Scratch

While there are many parameters making a game unique e.g. story, look or game-flow, some components in games are used by nearly all games independent from their genre (e.g. the use of a finite state machine for its logic or the need to map an item to a field) [28]. So, we tried to gather those common components in the modules of *Gamework* to allow a designer to easily reuse them. In our case, these modules include a management system for game items, a login system or a graphical editor for connecting virtual objects to coordinates or manipulating them, just to name a few.

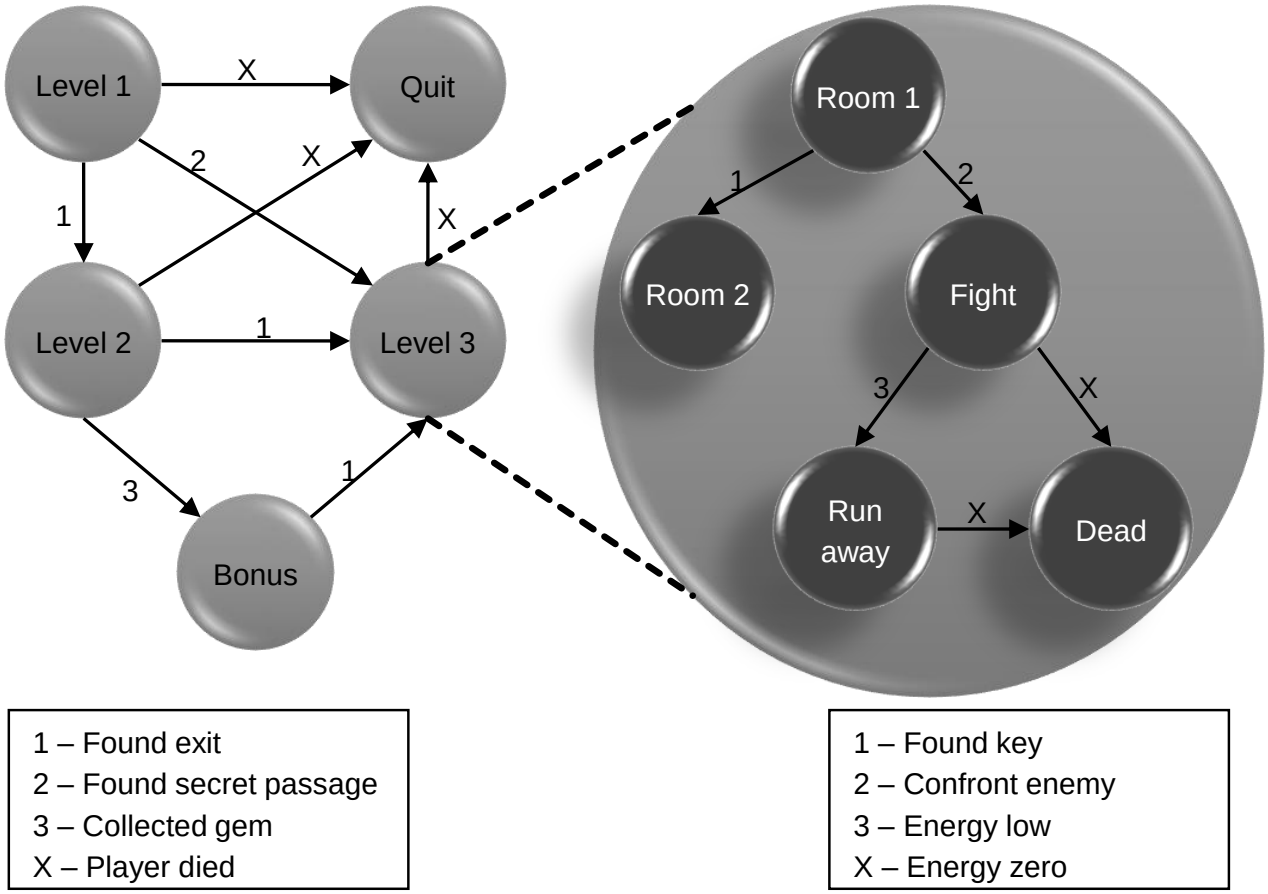

Figure. 3: Example of a FSM for the level order (left) and the sequence of action within a level (right)

Figure 4 shows an example how to create a new game from scratch while reusing existing components. In this use-case the game designer combines the content originated in Game X, the interfaces designed in Game Y and the login system of Game Z. In order to receive a runnable game she only has to add some logic to those preconfigured components.

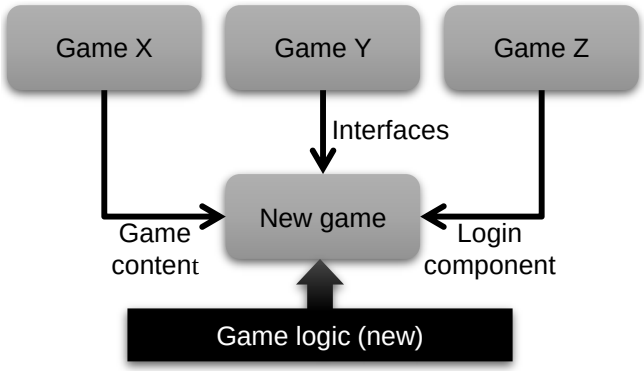

Figure. 4: Use-case how to create a game from scratch

In a framework for non-pervasive games, *Gamework*'s packages would probably differ slightly. Due to the strictly modular design of *Gamework*, a designer will still have to create the unique parts of her game only, while she can include the common ones from *Gamework*'s packages.

5) Summary

Table 1 summarizes the main features regarding the support given by *Gamework*, requirements towards each game, programming skills that are needed and finally the customization potential.

B. Content and Game Data Management

Alan Demers, et al. [29] state that a proper data management system is essential for the success of a game. Further studies revealed that the game's genre is irrelevant: It does not matter if a game uses a database directly, such as any arbitrary MMOs (massively multiplayer online games) such as *World of Warcraft* [30], or whether the data management is realized by an independent solution [31]. While dealing with the game content a huge amount of data has to be processed in any case. Hence, Mark Deloura [32] introduces the so called data-driven game design. In this development technique game content is separated from game logic. This technique has two huge benefits: On the one hand, the interest in a game can be increased in a major way, as anybody can add new game content. On the other hand, analyzing the game data becomes possible, as it is stored at a separated location. Such analysis methods are well-known in other database sectors, e.g. data warehousing. Remi Lehn et al. [33] discuss various types of analysis tools available in that scope. Nevertheless, such tools are hardly used for game data and content, although its value could be incalculable. We will present a possibility how to

Table 1: Comparison of the four customization classes.

Context in games	**User-generated content**
• Geo functions are provided • Game type has to be applicable • No programming skills needed at all • Poor customization potential	• WYSIWYG-editor for new content • Game must use content management system • Player has to deal with editor • Moderate customization potential
Game-flow adaption	**Creation from scratch**
• Game-flow modeled as an FSM • Game must use the FSM module • Player has to model the game-flow • Great customization potential	• Reuse-oriented framework design • Game is not bound to any limitations • Player has to be skilled in programming • Virtually unlimited customization potential

use it in Section V. For these reasons, we follow Deloura's approach in *Gamework*.

C. Platform Independency

Since there are at least three different key players in the Smartphone market (Apple, Google and Nokia) a platform dependent solution for each of them would neither be neither satisfying nor efficient. Therefore, we propose to implement pervasive games as web applications according to Jonathan Stark [34]. The HTML5 standard [35] features support for the use of context data within websites. In this way any mobile device with a modern browser (such as Google Chrome) can be used to play the games created with *Gamework*.
A pleasant side effect, which arises automatically by this decision, is the fact that installing the game will be unnecessary. Thus, more people are willing to test a game as they do not have to expect any consequences such as remaining data, wasted memory or complex removal procedures.

III. *Gamework* Architecture

The *Gamework* approach is based on a framework for mobile location-based browser games [19]. Since we want to reach as many users as possible, we cannot rely on high requirements towards the mobile device. Therefore, *Gamework* uses a thin client realizing the context date retrieval and the presentation issues, only. The hardware-heavy processing of the game logic as well as the management of all game-related content is done by a central server. As the amount of game data could get extremely huge, while – as stated in Subsection II-B – a proper data management is essential for the success of a game, it cannot be held on the mobile device. These considerations correspond to the well-known three tier architecture with a strictly separated client, logic and data layer.
Figure 5 depicts this architecture and the most prominent modules of *Gamework* as well as their associated customization techniques.

The main component of *Gamework* is the game item management module. It provides a generic class definition for game items in general as well as expandable predefined class definitions for frequently required data-types such as 'Single-player' or 'Money'. New subclasses (e.g. 'Team', 'Barrier', etc.) can easily be modeled by listing their features, such as a name, a value or some measurements, while attributes and behaviors appropriate for all types of game items are inherited from the generic superclass. Concrete game items are created as instances of these classes. A WYSIWYG-editor for the creation of user-generated content is able to support the creation of all items whose class definitions are available. As all other framework components communicate via the game item management module, it has also to forward all context or input data accruing on the mobile device to the context data handling module. Methods such as calculating relative movements from absolute positions or placing the player's avatar on the game field corresponding to her current position are provided by this module. While the mobile device is able to determine absolute positioning data (e.g. GPS data) only, these methods are required in order to map these coordinates to an in-game location or calculate a player's relative movement from this data in order to enable context-driven adaption. On top of the game item maintenance module, a finite state machine, using *Gamework*'s game item definitions, is available to model all game states. The coarse sequence of events – e.g. login $\rightarrow$ start $\rightarrow \ldots \rightarrow$ end – is defined by this module. Corresponding to the paragraph about game-flow adaption in SubsectionII-A, the different stages of a game and even every single task or the actions currently available within these stages can be set by defining some states and transitions. Any modification here has an immediate impact on the game-flow. Such a modification can be realized by manipulating an XML-like file manually. An editor for finite state machines with a graphical interface, as there are many examples available (e.g. [36]), could easily support the customization of the game-flow even further.
Finally, the game item management module also provides methods to make any data persistent when transferring it to the data layer. Thereby, all relevant game data can be saved or loaded in order to restore former game states. So, long lasting and even infinite continuous games in persistent virtual worlds can be realized.
Since any modules will work independently from each other, single modules can be used separately for the development of a game, while the unneeded remaining components of *Gamework* do not have to be used. The reuse-oriented development methodology guarantees that a developer will not have to reinvent the wheel over and over again and will boost the game development process [37].

IV. Sample Applications

In order to demonstrate the capabilities of *Gamework*, we implemented two pervasive games with *Gamework*: *1-2-3...* and P^2 *– Pervasive Pairs.1-2-3...* is a conversion of a geocaching-like search game where virtual items are placed on a map and the player has to collect them. P^2 is a location based version of the well-known tabletop game *Memory*®by Ravensburger™. We will describe them more detailed in the following subsections.

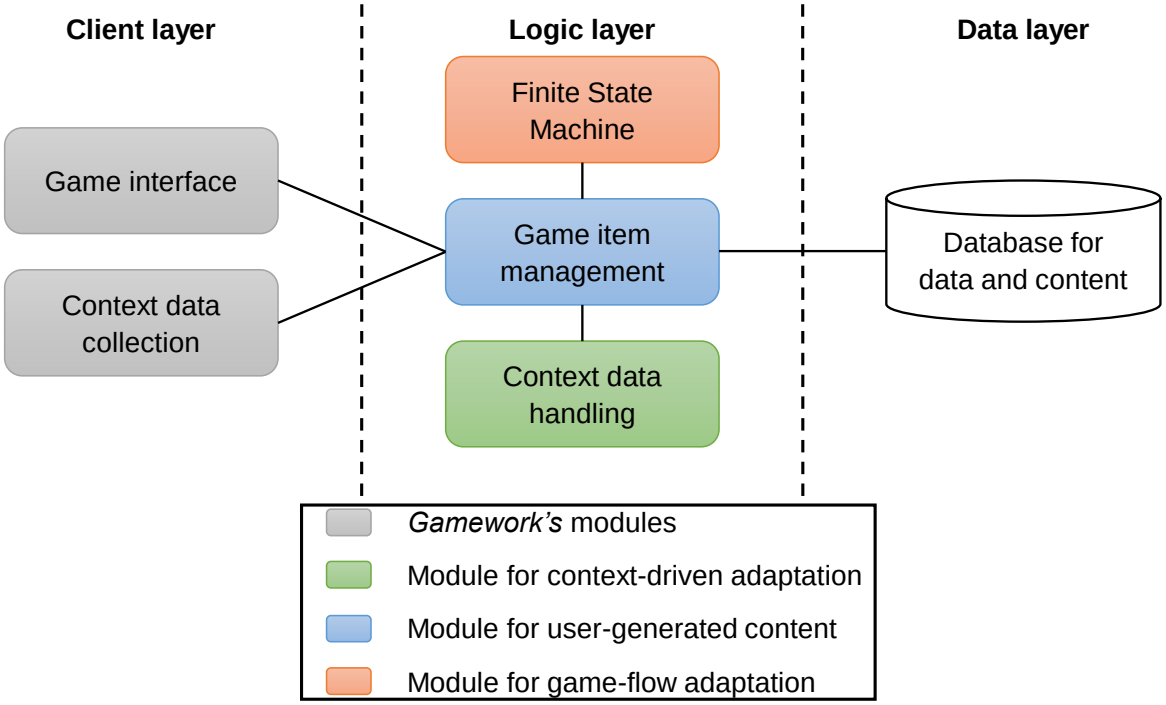

Figure. 5: The architecture of *Gamework*

A. 1-2-3...

1-2-3... is a location-based game for Android Smartphones. A player gets the quest to search for a hidden virtual item somewhere in her surroundings. No further details about the location of the cache are given, but the coordinates of a place where additional information can be acquired. This can be accomplished either by simply reaching that location – maybe within a given amount of time – or by answering a multiple-choice-question that has something to do with the certain location. According to the outcome of this task further hints about the final destination are revealed. E.g. by answering a question completely perfect the right coordinates are given while a not quite so good answer will lead to a detour.
Each of these quests is modeled as a finite state machine with a separated state for all locations. Since any inter task has one or more following task (one per "quality level" describing how good a player has solved her task) they represent the inner states of the finite state, while the final destination is our finite state. The defined outcomes of each task are modeled as conditions for the transition. Thereby, the player gets an impression on how to use and configure the finite state machine.
1-2-3... can be customized in two ways: On the one hand it automatically adapts the map coordinating with the current position of the player. Thus, only quests within the reach of the player are offered to her. On the other hand new quests can be created or existing quests can be expanded. Figure 6 shows an example how to create a quest for *1-2-3...* with the help of the FSM. When a player reaches the quest entrance (yellow smiley) this quest gets active. The first task is a multiple-choice-question task (red ring). If the player can answer the question right, then she is guided to location $1A$ and has to follow track 'A' (green dots) henceforth. If the answer has been wrong, then she has to do the detour 'B' with more tasks (blue dots). Task $2B$ is a time challenge (implied by the hourglass). When she reaches the location $2B$ too slow, then she has lost and the game is over. Finally, when the player reaches the goal (red X), she has won and her time is recorded, regardless of which track she came here. On the right side of Figure 6 the corresponding FSM is shown.
After arranging all states and configuring any condition the creator can invite a friend to test the new quest. It should be noted that the creator does not necessarily need to be at the site of the action – neither when the quest gets accomplished, nor at creation time! So anybody can hide an item virtually anywhere and anytime and send the quest to some of her friends far away.

B. P^2 – Pervasive Pairs

A more complex game created with *Gamework* is P^2 – *Pervasive Pairs*. The main concept of P^2 is based upon the card game *Memory*®by Ravensburger™. In a '*Pairs*'-game – also known as Concentration – a certain amount of cards are laid face down on a table. Two (or more) players have to collect as many of those cards as possible. To collect a card, a player has to find two cards showing the same image. Hence, in turns, each player chooses two cards and turns them face up. If their images do not match, she has to turn them face down again. The game ends when the last pair has been picked up. Then, the player who has collected the most pairs has won the game. We had to modify some of those rule in order to create a location-based version of Pairs. First of all, we replaced the real cards by virtual ones. Thus, a player does not have to use the default card images, as they can be exchanged easily. Furthermore, neither their amount nor their size is set. Unlike Pairs the cards are not laid on a table but they are randomly

Figure. 6: How to create a quest for *1-2-3. . .*

spread in the player's surroundings. In order to turn a card face up, a player has to reach the center of the card. As a card could be placed in an unreachable region (such as private property or natural obstacle) a certain tolerance range around the card's center can be granted. Then, this card belongs to the player and is locked for everyone else until she has selected a second one. In P^2 this additional rule becomes necessary, as all players make their moves simultaneously. This is our second big extension to the original rule-set. Test runs showed that it is beneficial to the gameplay if all players are always active and do not have to wait for the other players. Without this rule modification all players would have to stay at their current position for most of the time and get bored. So, not only memorize abilities determine who will win the game, but also the individual fitness level of each player. When two cards are locked for one player, their values have to be checked. In contrast to *Pairs*, a positive match does not necessarily have to mean that the cards show the same image. One could think about arithmetic expressions on the half of the cards and the corresponding results on the other half. Another possible scenario could provide pictures of famous football players and their team logos which have to be found in order to score.

To ensure that there is always the largest possible number of other player available, they must not necessarily be at the same location. The same virtual card can be at location A for player 1, while it can be found at location B for player 2, as all cards are randomly spread anywhere in the surrounding anyway. Solely, it has to be ensured that the distances and the relative positioning of the cards are equal for all players for reasons of fairness. Nevertheless, there can be different difficulty levels, as the terrain can be more or less hostile.

As *Gamework* should enable a player to customize a game as far as possible, P^2 also provides various adaptation options. First of all, the cards are spread in the player's surroundings, no matter where she starts the game without any interventions necessary, as described in the last paragraph. Secondly, a player can replace all images of the cards by any image or text given in a supported format. And finally, the game logic can be changed, so that a positive match does not necessarily mean that the cards' images have to be equal, but that some values correspond to each other.

Figure 7 shows two different instances of P^2. The player on the left side (blue arrows) has chosen drawings of objects from the CS area as images for her cards. The player on the right side (green arrows) made more customizations: She did not only change the theme of the cards sport area, but she also changed the game logic. Now a player has to find a picture of a ball has to recognize the corresponding game and finally has to find a card with the game's name on it. Thereby, an educational aspect comes into the game.

Figure. 7: Examples for two different customizations to P^2

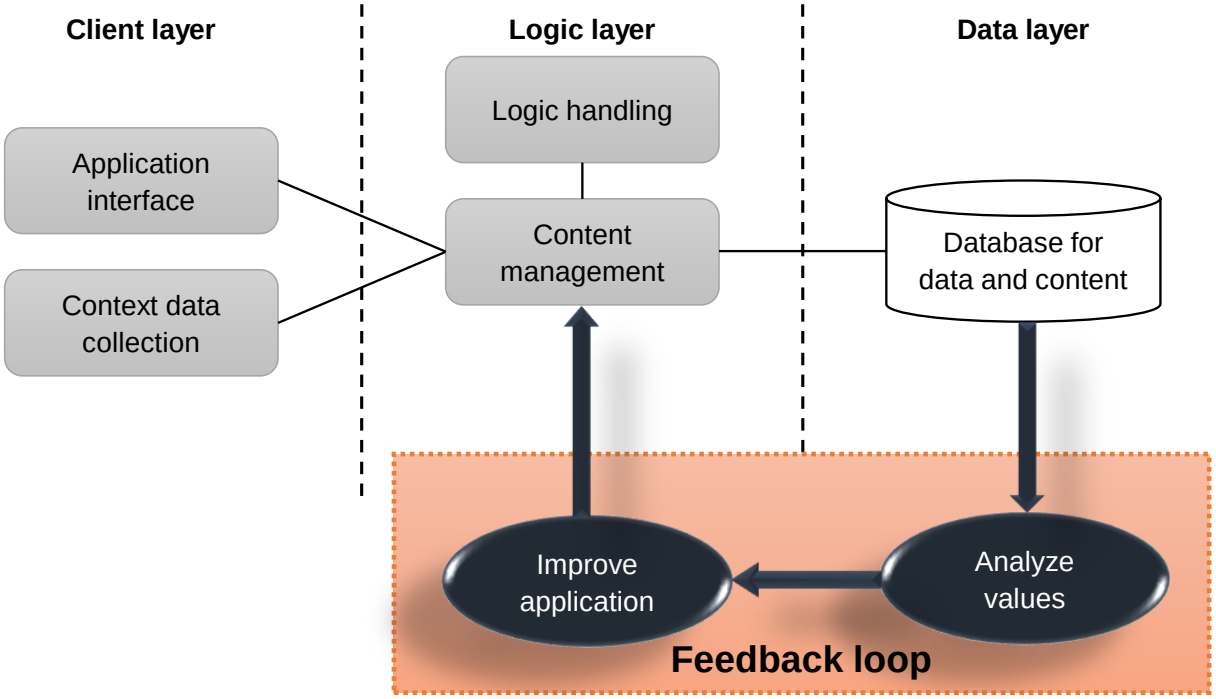

Figure. 8: Generalized and extended architecture of *Gamework*

V. Future Work

Even though the spotlight of *Gamework* has always been on the area of mobile context-aware games – primarily it has been developed as a possibility for young children and adolescents to learn programming step by step with quickly visible outcomes in a field they like – there is a clear perception that it carries over to mobile and context-aware applications in general. We use the application area of games as a kind of test bed for the development of new and innovative techniques for customizable mobile location-based applications and services as well as for gaining usage experience and assessments on these techniques easily. Similar approaches have long be known and are well-accepted in other Computer Science research areas, e.g., using soccer playing robots in artificial intelligence as the base for research on team behavior, strategies or optimization [38].

As in any application privacy is always a matter of concern, we cannot disregard this topic in *Gamework*. Therefore, we will have to implement mechanisms to assure the protection of the user's or application's context data. Although privacy is indeed important in the area of games, it is a must in other areas such as healthcare. For this reason, we will have to consider different approaches, e.g. Arijit Ukil's scheme focusing on protecting spatial and temporal contextual information [39] or Simon Moncrieff et al.'s formal methodology for designing privacy mechanisms [40].

After the extension of *Gamework* to a more general one supporting a broader class of web-applications we will add a feedback loop with an analysis concept in order to enhance user acceptance, usability and effectiveness, as depicted in Figure 8 [41]. The feedback loop will introduce self-improvement capabilities: Since all client and application data are available in the content management component and at the data layer, customer-specific as well as cohort-specific or crowd-specific analyses become viable in order to improve application or service quality. Depending on the concrete kind of application, the operating mode and methodology of the feedback loop will differ.

Therefore, novel techniques have to be devised to customize the feedback loop according to what data has to be analyzed, which analysis methods are applied, and what effect the results on the application have. E.g. a quiz could automatically give additional hints for a particular question, when a lot of requests are detected or a navigation system giving recommendations which see-sights a tourist should have seen, depending on preferences of similar users.

With *vHike* [42], a hitchhiking management system, we already introduced a general application which benefits of the existing parts of our enhanced version of *Gamework*.

VI. Conclusion

In this paper, we present *Gamework*, a framework for customizable mobile location-based games as a specialization for customizable mobile context-aware applications and services. Therefore, we classified different customization techniques for pervasive games. Each of these classes is characterized regarding to the adaption potential of it, the required programming expertise and how this method can be supported by *Gamework*. A player may adapt any game created with *Gamework* by changing the context, adding user-generated content, modifying the game-flow or implementing new games from scratch by reusing existing modules of *Gamework* or implementing new ones, according to her programming skills. In order to test the advantage of our approach, we present two prototypes of location-based mobile games which were implemented with *Gamework*.

Furthermore, we investigate basic technologies, gain experiences via evaluation of a prototype implementation and finally transfer the results back to the more general area of mobile and context-aware applications and services. Our strategy is to exploit the application area of games as a kind of test bed for the development of new and innovative techniques towards customizable mobile and context-aware applications and services.

In addition to create a user-friendly, easy understandable, ergonomic design for *Gamework*, we identified the feedback loop as the major challenge for future work. So we are aware of the fact that we either will be able to develop (auto-) adaptive solutions for given use cases, only or novel techniques have to be devised to customize the feedback loop itself. Nevertheless, our approach will be able to make a contribution not only to create new pervasive services but also to improve existing ones.

References

[1] I. Bokun and K. Zielinski, "Active Badges – The Next Generation," *Linux Journal*, vol. 54, pp. 24–26, 28–29, 1998.

[2] M. Weiser, "The computer for the 21st century," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 3, pp. 3–11, 1999.

[3] S. Lohr, "Body Sensing Comes to Smartphones," New York Times (Bits Blog on www.nytimes.com), October 2010. [Online]. Available: http://bits.blogs.nytimes.com/2010/10/11/body-sensing-comes-to-smartphones/

[4] A. Ahlund, "iPhone App Sales, Exposed," TechCrunch (Tweet on techcrunch.com), May 2010. [Online]. Available: http://techcrunch.com/2010/05/16/iphone-app-sales-exposed/

[5] C. Magerkurth, A. D. Cheok, R. L. Mandryk, and T. Nilsen, "Pervasive games: bringing computer entertainment back to the real world," *Computers in Entertainment*, vol. 3, pp. 1–19, 2005.

[6] W. Broll, J. Ohlenburg, I. Lindt, I. Herbst, and A.-K. Braun, "Meeting technology challenges of pervasive augmented reality games," in *NetGames '06: Proceedings of 5th ACM SIGCOMM Workshop on Network and System Support for Games*. New York, NY, USA: ACM, 2006.

[7] V. Paelke, C. Reimann, and D. Stichling, "Foot-based mobile interaction with games," in *ACE '04: Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology*. New York, NY, USA: ACM, 2004.

[8] A. D. Cheok, A. Sreekumar, C. Lei, and L. N. Thang, "Capture the Flag: Mixed-Reality Social Gaming with Smart Phones," *IEEE Pervasive Computing*, vol. 5, pp. 62–69, 2006.

[9] R. T. Azuma, "A Survey of Augmented Reality," *Presence: Teleoperators and Virtual Environments*, vol. 6, pp. 355–385, 1997.

[10] R. Wetzel, R. McCall, A.-K. Braun, and W. Broll, "Guidelines for designing augmented reality games," in *Future Play '08: Proceedings of the 2008 Conference on Future Play: Research, Play, Share*. New York, NY, USA: ACM, 2008.

[11] A. D. Cheok, K. H. Goh, W. Liu, F. Farbiz, S. W. Fong, S. L. Teo, Y. Li, and X. Yang, "Human Pacman: A mobile, wide-area entertainment system based on physical, social, and ubiquitous computing," *Journal of Personal Ubiquitous Computing*, vol. 8, pp. 71–81, 2004.

[12] R. A. Ballagas, S. G. Kratz, J. Borchers, E. Yu, S. P. Walz, C. O. Fuhr, L. Hovestadt, and M. Tann, "REXplorer: a mobile, pervasive spell-casting game for tourists," in *CHI '07: CHI '07 extended abstracts on Human factors in computing systems*. New York, NY, USA: ACM, 2007.

[13] C. Stach, "Gamework – A customizable framework for pervasive games," in *GET '10: Proceedings of the IADIS International Conference Game and Entertainment Technologies 2010*. Freiburg, Germany: IADIS Press, 2010.

[14] S. P. Hall and E. Anderson, "Operating systems for mobile computing," *Journal of Computing Sciences in Colleges*, vol. 25, pp. 64–71, 2009.

[15] H. Eirund, B. Grüter, and A. Mielke, "Der Spieler macht das Spiel - Mechanismen der Autorenrolle in mobilen Spielen," in *Tagungsband der 34. GI-Jahrestagung*, ser. Lecture Notes in Informatics. Springer, 2004.

[16] S. Wolff and B. Grüter, "Context, emergent game play and the mobile gamer as producer," in *Tagungsband der 39. GI-Jahrestagung*, ser. Lecture Notes in Informatics. Springer, 2008.

[17] J.-P. Tutzschke and O. Zukunft, "FRAP: a framework for pervasive games," in *EICS '09: Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*. New York, NY, USA: ACM, 2009.

[18] G. Hong, H. Trætterberg, A. I. Wand, and M. Zhu, "TeMPS: A Conceptual Framework for Pervasive and Social Games," in *Proceedings of the Third IEEE International Conference on Digital Games and Intelligent Toy Enhanced Learning*. Washington, DC, USA: IEEE Computer Society, 2010.

[19] A. Brodt and C. Stach, "Mobile ortsbasierte Browserspiele," in *Tagungsband der 39. GI-Jahrestagung*, ser. Lecture Notes in Informatics. Springer, 2009.

[20] C. Stach, "Mobile ortsbasierte Browserspiele," Master's thesis, University of Stuttgart : Center of Excellence SFB 627 (Nexus: Spatial World Models for Mobile Context-Aware Applications), Germany, 2009.

[21] R. Hunicke, "The case for dynamic difficulty adjustment in games," in *ACE '05: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*. New York, NY, USA: ACM, 2005.

[22] C. Schlieder, P. Kiefer, and S. Matyas, "Geogames: Designing Location-Based Games from Classic Board Games," *IEEE Intelligent Systems*, vol. 21, pp. 40–46, 2006.

[23] M. Schmatz, K. Henke, C. Türck, C. Mohr, and T. Sackmann, "SYGo - a location-based game adapted from the board game Scotland Yard," in *Tagungsband der 39. GI-Jahrestagung*, ser. Lecture Notes in Informatics. Springer, 2009.

[24] Linden Labs, "Linden Scripting Language," Second Life LSL Portal, February 2011. [Online]. Available: http://wiki.secondlife.com/wiki/LSL_Portal

[25] J. Krumm, N. Davies, and C. Narayanaswami, "User-Generated Content," *IEEE Pervasive Computing*, vol. 7, pp. 10–11, 2008.

[26] P. Sweetser and P. Wyeth, "GameFlow: a model for evaluating player enjoyment in games," *Computers in Entertainment*, vol. 3, pp. 1–24, 2005.

[27] M. J. Taylor, D. Gresty, and M. Baskett, "Computer game-flow design," *Computers in Entertainment*, vol. 4, pp. 1–10, 2006.

[28] A. Rollings and E. Adams, *Andrew Rollings and Ernest Adams on Game Design*. New Riders Games, 2003.

[29] A. Demers, J. Gehrke, C. Koch, B. Sowell, and W. White, "Database research in computer games," in *SIGMOD '09: Proceedings of the 35th SIGMOD international conference on Management of data*. New York, NY, USA: ACM, 2009.

[30] Blizzard Entertainment, "World of Warcraft," Battle.net, March 2011. [Online]. Available: http://us.battle.net/wow/en/

[31] W. White, C. Koch, N. Gupta, J. Gehrke, and A. Demers, "Database research opportunities in computer games," *ACM SIGMOD Record*, vol. 36, pp. 7–13, 2007.

[32] M. Deloura, *Game Programming Gems*, D. Treglia, Ed. Rockland, MA, USA: Charles River Media, Inc., 2002.

[33] R. Lehn, V. Lambert, and M.-P. Nachouki, "Data warehousing tool's architecture : From multidimensional analysis to data mining," in *DEXA '97: Proceedings of the 8th International Workshop on Database and Expert System Applications*. Washington, DC, USA: IEEE Computer Society, 1997.

[34] J. Stark, *Building Android Apps with HTML, CSS, and JavaScript: Making Native Apps with Standards-Based Web Tools*, B. Jepson, Ed. Beijing and Cambridge and Farnham and Köln and Sebastopol and Tokyo: O'Reilly Media Inc., 2010.

[35] W3C, "HTML5," Editor's Draft 14 February 2011, February 2011. [Online]. Available: http://dev.w3.org/html5/spec/

[36] A. Darovsky, "Finite State Machine Editor (FSME)," FSME Project Page, July 2009. [Online]. Available: http://fsme.sourceforge.net/

[37] B. Neto, L. Fernandes, C. Werner, and J. M. de Souza, "Reuse in Digital Game Development," in *ICUT '09: Proceedings of the 4th International Conference on Ubiquitous Information Technologies & Applications*. Washington, DC, USA: IEEE Computer Society, 2009.

[38] U.-P. Käppeler, O. Zweigle, K. Häußermann, H. Rajaie, A. Tamke, A. Koch, B. Eckstein, F. Aichele, D. DiMarco, A. Berthelot, T. Walter, and P. Levi, "1.RFC Stuttgart Team Description 2010," in *RoboCup Team Description Papers 2010*, RoboCup, Ed., Singapur, 2010.

[39] A. Ukil, "Context Protecting Privacy Preservation in Ubiquitous Computing," *International Journal of Computer Information Systems and Industial Management Applications*, vol. 3, pp. 228–235, 2011.

[40] S. Moncrieff, S. Venkatesh, and G. West, "A Framework for the design of privacy preserving pervasive healthcare," in *ICME'09: IEEE International Conference on Multimedia and Expo*. Piscataway, NJ, USA: IEEE Press, 2009.

[41] C. Stach, "Gamework – A customizable framework for pervasive games (Doctoral Colloquium)," in *ICPS '10: Proceedings of the 7th International Conference on Pervasive Services*. New York, NY, USA: ACM, 2010.

[42] ——, "Saving time, money and the environment – vHike a dynamic ride-sharing service for mobile devices," in *PERCOM '11: Proceedings of the 9th IEEE International Conference on Pervasive Computing and Communications*. Washington, DC, USA: IEEE Computer Society, 2011.

Author Biographies

Christoph Stach is a Research Assistant at the Institute of Parallel and Distributed Systems, University of Stuttgart in Germany. He received his Diploma for his work on mobile location-based browser games from the Applications of Parallel and Distributed Systems Department, University of Stuttgart in 2009. Currently he is doing his PhD at the University of Stuttgart. His main areas of interest are pervasive computing and customizable mobile Web-services. In this area he already published various papers on different conferences. On top of that he organized and arranged a lot of workshops dealing with programming paradigms for mobile and context-based computer games.