fAARS:
A Platform for Location-Aware Trans-reality Games

Lucio Gutierrez, Eleni Stroulia, Ioanis Nikolaidis

Computing Science, University of Alberta, 2-21 Athabasca Hall, Edmonton, AB T6G 2E8, Canada
{lucio, stroulia, nikolaidis}@ualberta.ca

Abstract. Gaming technology has been advancing in leaps-and-bounds. Users can easily and intuitively record their real-world experiences through mobile devices, and commodity virtual worlds enable users from around the world to socialize in the context of realistic environments where they simulate real-world activities. This synergy of technological advances makes the design and implementation of trans-reality games, i.e., games that blend the boundaries of the real and virtual worlds, a compelling software-engineering problem. In this paper, we describe fAARS, a platform for developing and deploying trans-reality games that cut across the real and parallel virtual worlds, offering users a range of game-play modalities. We place fAARS in the context of recent related work, and we demonstrate its capabilities by discussing two different games developed on it, one with three different variants. Finally, we briefly discuss our experience with these games.

Keywords. Game platform; trans-reality games; virtual worlds; mobile games

1 Introduction

Location-based games are a particular type of pervasive games that use the physical space of our entire world as a game board [1]. Their implementation relies on special-purpose middleware that integrates the players' smart-phones, so that the players can share with each other information abut their location, surroundings and actions. The game engine interprets this information in the context of the basic game story line and progresses the game narrative. The social engagement of these games and the increased programmability and adoption of smart-phones that offer seamless connectivity across WiFi, 3G or 4G networks at affordable service plans [2] are making these games increasingly popular.

The players' game experience can be further enhanced with augmented-reality applications that enable them to see virtual content projected on top of physical objects through the screen of their smart-phones. In addition, by setting up parallel virtual worlds, the players can experience the game in the real world and in alternate realities, possibly at the same time. This technological convergence, along with the current "gamification" of services, has brought a new breed of pervasive games called "trans-

reality games". This term refers to a relatively new, yet increasingly interesting, class of games, played using mobile devices in outdoor and indoor spaces, and in virtual worlds. Trans-reality games were first conceptualized by Lindley [3] in 2004. This type of games can be deployed and played in "natural" neighborhoods, such as a campus, a city, or a broader geographical area. Game players interact with the games (a) in the real world through location-specific clues communicated to them through smart-phones that "augment reality" and are aware of the player's locations using GPS or QR codes, and (b) in (possibly multiple) parallel virtual world(s), which represent alternate reality(ies) that reflect the real world in some dimensions and extend it in others. The development of a platform for authoring and deploying trans-reality games has been the objective of our work with fAARS.

The fAARS (for Augmented Alternate Reality Services) platform supports the implementation and deployment of trans-reality games in the virtual treasure-hunt and role-playing style. fAARS consists of a set of components, flexibly integrated through RESTful APIs wrapping its event-driven game engine. So far, we have developed two different trans-reality games using the fAARS platform. The diversity of these two games provides strong supporting evidence for the generality and usefulness of fAARS as a trans-reality game development platform. The first game was the "Human Geometric Orientation" game. Through this game, players in a virtual world participate in an experiment, designed to study human orientation mechanisms. The second game, "Outbreak: Safety First", is a serious game designed to educate health-science students in precautionary procedures designed to avoid nosocomial infections, and can be played in three different game-play modalities. These games are quite different from each other and one was deployed in three different variants. Also important to note is that the specification of both of them originated outside our group, with people who were not aware of the features of fAARS. As such, these two games constitute strong evidence of independent validation of the broad scope and usefulness of fAARS. These two games resulted in interesting results, demonstrating the value of the platform as a tool for simulation and education.

In the rest of this paper we discuss our work and experience to date. In section 2, we review the background for our work, by reviewing recent trans-reality games and existing platforms for location-aware games. In section 3, the design and implementation of fAARS and its constituent components are fully described. In section 4, we report on our evaluation of the platform through the development of two separate games. Finally, in section 5, we conclude with the contribution of our work with fAARS and we discuss our plans for further development of functionality in the fAARS platform and future trans-reality games we are planning to support.

2 Trans-reality Games and Pervasive-Game Platforms

The "pervasive games" heading includes a broad range of games that integrate ubiquitous technology with physical spaces to provide a gaming experience. Montola [4] introduced a conceptual framework for analyzing the spatial, social, and temporal features of pervasive games, leading to a classification of these games into six types.

Mixed-reality games are played in physical spaces where virtual and physical components co-exist. *Trans-reality games* are played in physical and virtual spaces at the same time, with actions of player avatars in the virtual world affecting the game play of players in the real world and vice versa. *Adaptronic games* are played in a virtual reality but react to real-world events, such as temperature or light-intensity changes. *Crossmedia games* involve different types of pervasive technologies. *Alternate-reality games* create the illusion that game-play events should be considered "real", thus creating an alternate reality where players act according to hidden clues. Finally, in *reality games*, everything that happens in the game is real and can have consequences on the real life of players.

2.1 Trans-Reality Games

Trans-reality games are a particularly interesting and challenging class of pervasive games. "ARQuake" [5] was one of the first outdoor/indoor augmented-reality location-aware games based on the desktop game Quake. Two years later, "Human Pacman" [6] was the first outdoor augmented-reality game to blend the real and virtual dimensions. Both games required players to wear their particular versions of cumbersome equipment. In 2005, "Magic Land" [7] blended 3D avatars of real players with 3D computer-generated animations, fusing the real and the virtual worlds in a novel way. Between 2004 and 2006, trans-reality games, such as "Can you see me now?" [8], "Uncle Roy All Around You" [9], and "I Like Frank" [10], demonstrated the use of mobile devices in real-world playgrounds blurred with virtual worlds, with players in both worlds advancing the game, although the actions allowed to players in the virtual world were very limited.

The above games were developed as one-of exercises in blending specific hardware and software technologies for the sake of their game narrative and cannot be reused. Recently though, with the increased adoption and standardization, of mobile platforms and the availability of commodity virtual worlds, the opportunity arises for developing more general platforms for a new breed of trans-reality games. "The Timewarp" [11] time-travel game combines virtual and physical spaces during gameplay. Players in the real world explore the past by displaying media content from a parallel virtual world on physical objects around them. They also have the opportunity to visit the past in the virtual world trough wearable computing. More recently, in "Alien Contact!" [12], players have to find clues in the real world by solving challenges in the form of questions, and looking for virtual characters in a parallel virtual world using handheld computers.

2.2 Platforms for Pervasive Games

In the above trans-reality games, virtual worlds have been used primarily as an information resource, and there is still substantial work to be done towards the comprehensive integration of virtual and real-world game-play. In developing the fAARS platform, we have aimed to support the development of a particular genre of location-based trans-reality games, where (a) the real-world space and locations (and any in-

formation associated with them) are mapped to the virtual world; (b) the movement and actions of the real-world players is reflected through their avatars in the virtual world; (c) the movement and actions of the virtual-world players is communicated to real-world players through their smart phones; and (d) in addition to real-world and virtual-world players, non-playing characters, controlled by a simulation engine, can also change the game state. As fAARS is on the overlap between trans-reality and location-aware mobile games, we review below the features of some important platforms for location-aware games.

fAR-Play [13] supports virtual treasure-hunt games with question challenges as part of the game-play. It supports indoors(outdoors) game play through QR codes (GPS). It uses Layar to provide augmented-reality content, maps to locate the treasures, and a mobile web site to provide feedback to players. It includes an authoring environment for rapid game prototyping. *SCVNGR* (htpp://www.scvngr.com) also virtual treasure-hunt games with a variety of challenge types, including taking pictures, decoding QR codes, and answering questions. It supports outdoors game play using GPS and maps as augmented-reality content. Player feedback is provided through a native iPhone and Android mobile applications. It also provides an authoring environment. *MUPE* [14] is an open-source platform for developing virtual treasure-hunt and RPG games on J2ME (http://www.oracle.com/technetwork/java/javame/index.html) phones. It supports outdoors game play and peer-to-peer player interactions through Bluetooth. Game authoring is supported through a set of configuration documents. *Mobile Chase* [15] supports the rapid prototyping of market-ready outdoor virtual treasure-hunt games for mobile phones that run the J2ME virtual machine. *FRAP* [16] supports the development of virtual treasure-hunt and capture-the-flag outdoor games, using GPS. Games advance through the collaboration among players who use a map as an augmented-reality feature that enables them to locate each other. *Wherigo* (http://www.wherigo.com) also supports games in the virtual treasure hunt style. It provides a game authoring application for developing games that can be downloaded to a mobile application that runs on GPS-enabled pocket PCs and Garmin (http://www.wherigo.com/garmin/default.aspx) handheld devices. It supports outdoors game play using GPS, and supports maps as part of the augmented reality feature of games. *PCAFPEA* [17] is a prototype framework for outdoors context-aware applications. It supports peer-to-peer interactions in games in the virtual treasure-hunt style, and offers a very interesting augmented-reality feature, namely 2D and 3D maps of the game space.

The comparison of the features of these platforms is summarized in **Table 1.** . As it can be seen in the table, a few of the platforms support indoors and outdoors game-play but none support virtual worlds; thus, they are not able to support trans-reality games. Most are tailored to individual game-play, except from FRAP that also supports teams. Most augment the players' game-play experience with game-related content; some include an augmented-reality browser that superimposes this content on the image that players see through their cameras, or on a map where they can see their (and other players') location(s). All support treasure-hunt games, with Mupe also supporting role-playing games and FRAP supporting capture-the-flag games. All support communication between players through the game engine, with Mupe and

PCFPA also supporting peer-to-peer interactions of players in close proximity to each other. Finally, most of the platforms offer some game-authoring tools.

Table 1. Pervasive-Games Platforms

	Spatial Support		Social/Collaboration Game Play		Forms of Augmented Reality			Game Domain			Communication		Game Authoring
PLATFORM	Indoors	Outdoors	Individual	Teams	Map	Augmented Browser	Raw Game Data	Treasure Hunt	RPG	Capture-the-flag	Peer-to-peer	Client-server	
fAR-Play	√	√	√			√	√	√				√	√
SCVNGR		√	√		√		√	√				√	√
MUPE		√	√				√	√	√		√	√	
Mobile Chase		√	√				√	√				√	
FRAP		√		√	√		√	√		√		√	
Wherigo		√	√		√		√	√				√	√
PCAFPEA		√	√		√	√	√	√			√	√	

3 Design and Implementation of fAARS

fAARS was conceived as a new-generation pervasive-game platform, extending the space of the game play from the real world (indoors and outdoors) to virtual worlds, thus supporting the development and deployment of location-aware trans-reality games. fAARS supports the deployment of trans-reality games in the virtual treasure-hunt and RPG (Role-playing Game) styles. fAARS integrates OpenSim (http://opensimulator.org/), an open-source implementation of Second Life (http://www.secondlife.com), the platform of the alternate virtual-world game-play. We chose OpenSim for three reasons. First, as an open-source platform, it is amenable to any desirable extensions. Second, as it replicates the look and feel of the popular Second Life virtual world, we anticipate an easier learning curve for game-play. Finally, OpenSim is already quite robust and functional, with a variety of tools to support the landscaping of spaces and the construction of building and object models, a feature essential in developing the games' alternate-reality environments.

Extending location-based games to virtual world gives rise to the task of defining a space for the virtual-world game-play. At this point, fAARS considers two types of mappings between the real-world and virtual-world spaces. In the "parallel worlds" mapping, the virtual-world setting mimics the real-world setting and the coordinates of every location in the real-world playground are transformed to a pair of virtual-

world coordinates. In this mode, the real and virtual worlds share the same conceptual metaphor, and every player location, whether in the real or in the virtual world, has a corresponding location in the virtual or real world. On the other hand, in the "points-of-interest" mapping, the virtual world may represent a conceptually different metaphor from the real world, and only specific points of interest may be mapped across the two. In this case, the players locations can only be reflected across the two worlds when the players are close to any of the pre-defined points of interest.

Given that it supports game play in two (at least) worlds, fAARS supports three different game-play modalities [18]. In the *Dormant* modality, players play the game in the real world using smart-phones. In the *Dreaming* modality, players play the game in the virtual world. Finally, in the *Astral-projection* modality, players choose to play in the virtual world or in the real world using smart-phones; in the latter case, the real-world players' avatars interact with other players and objects in the virtual world at the same time.

3.1 The fAARS Software Architecture

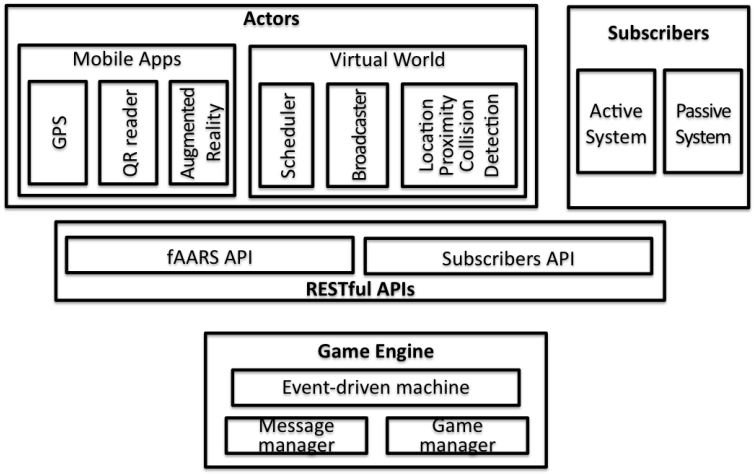

Fig. 1. The fAARS Platform Software Architecture

As shown in **Fig. 1.** , the fAARS platform consists of two main components: (a) the *Actors* (and their devices) corresponding to the real-world and virtual-world players; and (b) the *Game Engine*, which is responsible for recognizing the *Actors'* actions and inferring the next game state based on the game rules. The *Actors* access the game engine through the *fAARS API.* Furthermore, external components (i.e., Subscribers) can be integrated with the *Game Engine* through the *Subscribers API*. The fAARS platform is implemented according to the Event-Driven SOA (Service Oriented Architecture) style [19]. The fAARS components interact with each other through a well-defined RESTful interface that allows for their extension and/or reimplementation.

The Actors

In fAARS, every player in the real and virtual worlds is represented as an *Actor*. In the real world, *Actors* interact with the game using a mobile application running on a smart-phone. In the virtual world, they are equipped with smart-phone emulators.

The mobile application senses the *Actors'* location and actions, using the smart-phone GPS and a built-in QR-code reader. It is implemented using HTML5+CSS+Javascript, relying on the Phonegap framework (http://phonegap.com/) for cross-platform support.

In the virtual world, players have the same cross-platform application running on a HUD (Heads-Up Display). The HUD senses the virtual-world through collision, proximity, and location detectors. As shown in Figure 1, the virtual world also provides a *Scheduler* and a *Broadcaster*. The former supports the communication between the virtual world and the game engine, through a XML-RPC port, and maintains a set of global parameters related to the virtual-world game. The *Broadcaster* maintains a list of the active virtual-world objects and forwards them relevant communications from the *Game Engine*. All these components have been implemented in the Linden Scripting Language (http://wiki.secondlife.com/wiki/LSL_Portal).

The Game Engine

This component is responsible for maintaining and advancing the state of the *game*; it receives *Events* from the *Actors*, processes the game rules and accordingly updates the state of the game, and notifies the players and registered *Subscribers*. The internal components of the *Game Engine* are implemented in PHP (http://www.php.net), and its database is deployed on the MySQL database manager server.

The *Game Engine* consists of (a) the *Event-driven Machine*, (b) the *Message Manager*, and (c) the *Game Manager*. **Fig. 2.** depicts the run-time behavior of the engine. *Actors* generate events which are received by the *Message Manager* that forwards it to the *Event-driven Machine*. The *Event-driven Machine* communicates with the *Game Manager,* which queries the fAARS database for the relevant game rules[1]. Based on the rules, the *Event-driven Machine* updates the state of the virtual world and the state of the *Actors* and notifies the *Subscribers* about any changes and *Events* that occur during game-play.

The *Actors* are notified about game-state changes by the *Message Manager* using the *Long Polling* approach, which is a is a communication protocol where the server holds a request for as long as there is no information available for the client. When the request times out, or information is sent to the client, the client automatically sends a new request to the server.

This enables the *Game Engine* to communicate with the *Actors* on demand and to update the cross-platform mobile applications and the virtual-world HUDs by regularly pushing information about the game, without requiring players to explicitly requests updates from the *Game Engine*. The *Message Manager* has been partly implemented using the Javascript Framework MooTools (http://mootools.net/) and runs on the APE (Ajax Push Engine) Project Server (http://www.ape-project.org/) which al-

[1] Note that, as of today, fAARS does not offer an authoring environment; therefore, the game rules have to be stored directly in the database by someone familiar with the fAARS database design.

lows to subscribe any piece of software interested in interacting with internal and external pieces of the fAARS platform.

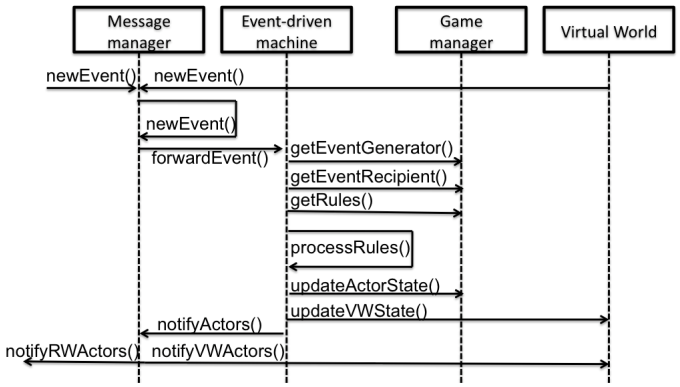

Fig. 2. Sequence Diagram of the Communication of the Internal Pieces in fAARS

The Subscribers RESTful Interface

fAARS provides a REST service that allows external systems to observe a game and to provide further downstream functionality to fAARS games, as if they were part of the platform. As shown in Figure 1, *Subscribers* can be either Active or Passive. Active subscribers can push events to the *Game Engine* where passive systems are only notified about *Events*. In either case, the external component needs to support the *Long Polling* messaging protocol and to understand JSON (http://www.json.org/) encoded messages. This service is composed of two main APIs.

The *registerObserver* API allows external systems to register as observers of the games running on fAARS. To that end, the system has to provide (a) a game-ID, identifying the game of interest; (b) a client-key, identifying the external system thus ensuring that only recognized subscribers can observe fAARS games; and (c) the desired *interactivity-mode*, specifying whether the external component will be interacting with fAARS in a passive or active mode. The *pushEvents* API allows external systems to push events into fAARS via real-time data streaming and requires four parameters: (a) the game-ID; (b) the client-key; (c) the event-type, i.e., the name of the *Event* that is being communicated to the Engine; (d) the event recipient identifying the *Actor(s)* who should receive it; and (e) up to two optional parameters, if the event type requires them.

The fAARS RESTful API

The functionality of every internal component of the *Game Engine* has been exposed as a set of RESTful APIs, to allow external and internal components of the platform to communicate with each other in a flexible manner. The fAARS RESTful API is composed of thirty APIs, which provide access to functionality of different internal pieces of fAARS, specifically to the internal pieces of the *Game Engine* as depicted in Figure 1. The APIs are divided in four categories, where each category represents a specific set of functions in fAARS; these are: a) event-driven machine (push events: 2 APIs), b) game manager (query the game state: 16 APIs), c) message manager (send

and receive messages: 2 APIs), and d) virtual world (update the virtual world: 10 APIs). These layer has been implemented in PHP, can be invoked via GET requests, and their response is provided in JSON formatted files.

Authoring fAARS Games Rules

fAARS does not have an authoring environment. The game rules are defined as *Events-Conditions-Actions* (ECA) triplets, stored over three database tables. The *Events* table specifies the event generator, its recipient, and its type. The *Condition* table specifies the name of a condition, the subject of the condition (generator or recipient), and the value of that condition, which can be the current state of an Actor in the game, the score of an Actor, or the location of an Actor or a group of Actors. Finally, the *Actions* table contains the name of the action, the recipient of the action, which may be the generator or the recipient of the *Event*, and the value of that action, which can be a score number, a location update in the real or the virtual world, or a state to which an Actor or group of Actors will transition.

3.2 Creating and Playing a Game in fAARS: An Illustrative Example

In this section, we describe the process of building a fAARS game with Pacman, the first game prototype developed in parallel with the development of fAARS. This game is an immensely popular arcade game that starred a small, yellow, puck-shaped character being chased around by ghosts while trying to eat every "pill" on a map.

The rules of this game are very simple. Pacman must attempt to collect all the "pills" and "powerpills" while avoiding being caught by four Ghosts chasing it. If Pacman can eat all the "pills" without being caught, then Pacman wins. Now, let us picture Pacman, but played on a real-life scale: city blocks are the map, Pacman and the Ghosts are players with smart-phones, and QR codes scattered throughout the game area are the "pills" that Pacman needs to scan in order to win the game.

Game development starts with the identification of the *Actors,* i.e., players. fAARS-Pacman has five *Actors*: one Pacman and four Ghosts. The rules of the game can then be specified. For example, the rule "A Ghost(generator) *captures* Pacman(recipient)" is represented as follows.

Event:	"A Ghost *collides* with Pacman"
Condition:	"When Ghost is in the *chasing* state" **and**
	"When Pacman is in the *running_away* state"
Actions:	(1) Change the current state of Pacman to "captured",
	(2) Update the scores of the Event generator and recipient.

Players walk around a specific physical area while decoding QR codes affixed to players and walls to advance the game, thus generating real world *Events*. In the context of the fAARS platform, an *Event* describes an interaction between two *Actors*. There can be multiple *Events* during game play, and they are all managed by the fAARS *Game Engine* in order to process the *ECA* rules to update the state of the *Actors* in the game. Furthermore, at any time during the game, players can track their scores and status on their smart-phones, and they can continue playing the game until either Pacman dies or when Pacman finishes eating all the "pills".

Once the game rules have been defined, the media content to be used in the mobile application (for real-world *Actors*) and the HUD (for virtual-world *Actors*) of the game is uploaded to the a fAARS repository. Finally, five player slots should be created in the database, each of them with a different game role; in this case either a Pacman or a Ghost.

At this point, the game has been specified and the mobile application and HUD are functional. If a virtual-world component is actually used, it will have to setup (the space landscaped and mapped to the real world and the objects modeled). Players can download the mobile application on their phones or login the virtual world to play using the HUD.

4 Our Experience with the fAARS Platform

We have validated the usefulness and generality of fAARS by developing two quite different games, both of which were defined by the needs of two different researchers at the University of Alberta. The "Human Geometric Orientation" game was designed by psychologists to evaluate subjects' strategies for orienting themselves within a space. The "Outbreak: Safety First" game was designed by an epidemiologist for training medical students in protocols for infection avoidance. These two games have generated very interested results in their corresponding areas of study, which however are outside the scope of this paper. In this paper, we review the games, their rules and their development process on fAARS.

4.1 The "Human Geometric Orientation" Game (HGO)

Animals and humans need to orient themselves in their environment, and their perception of the geometry of their space is an essential clue. Systematically examining whether and how each geometrical property impacts orientation implies the need to "build" environments in a controlled manner with their properties systematically varied. This is practically impossible and this is why our psychology colleagues decided to conduct their study by developing a fAARS game.

In this game, players were trained to locate two geometrically equivalent corners in a number of parallelogram-shaped rooms in a first-person navigable virtual world. The task could be solved based on identifying three features: angular amplitude, relative wall lengths, or a principal axis. In the training phase of the game, players had to reach a certain competency; next in the testing phase of the game, they were asked to accomplish the task in worlds where some of the features were eliminated. The impact to their performance was then used to infer which of the features are more important.

There are two important points to be made on the HGO game. First, the most important motivation for developing the game was that the researchers could systematically specify the types of room shapes they wanted, and we were able to systematically construct them in the virtual world using the OpenSim building tools. Second, since it was impossible to construct parallel real-world settings, the HGO game was

played only in the virtual world, in the so-called *Dreaming modality*, with players logging in the virtual world and interacting with the room corners to generate virtual-world *Events*.

Referring to **Fig. 1.** , the fAARS components used for the HGO game are the virtual world, the game engine, and an external system as an active subscriber.

Actors. A total of 99 *Actor* instances (one for each subject) with unique IDs were created to allow participants to interact with the corners in each of the rooms in the virtual world. The room corners were also represented as *Actor* instances, in order to rely on the virtual-world collision detection for generating corner-choice events

Game Engine. The HGO game has a single rule that simply gives points to a player when s/he chooses the right corner. The ECA triple describing this game rule involved (a) a corner-choice event, (b) no conditions, and (c) the action of increasing the score of the *Actor,* if the choice was correct.

Virtual World. As we have already discussed, the virtual world was the setting f the game with rooms constructed manually in a systematic manner and the corners associated with invisible *Actors*.

Event Subscribers. There is a single active subscriber that keeps track of the number of tries of every *Actor* for each of the rooms in the virtual world, and updating the location of the *Actor* instances based on the number of tries.

4.2 The "Outbreak:Safety First" Game (Outbreak)

The second game developed on fAARS was a serious game for educational purposes. "Outbreak: Safety First" was designed as a learning activity for medical students, taking an epidemiology curse. The game narrative places the players in a hospital where they have to visit a number of patients. As they move from one patient to another they have to avoid infectious viruses and to answer questions. At the patients' rooms, they have to follow appropriate precautionary procedures to protect themselves from infections, depending on the patient symptoms.

Taking full advantage of the fAARS functionalities, this game was deployed in three different variants. The first variant was played in the real world only, in a teaching-hospital ward. Players have to scan QR codes to see the symptoms of the patients they visit, to select the appropriate gear to don, to answer questions, and to get treated if they were infected. They become infected if they come close to an airborne virus or another infected player. The viruses, their infection cycle, their contagion to patients and players are controlled through a simple simulation in the virtual world, setup in a parallel mapping to the real world. As the game evolves, players "get infected" through actions dictated by the game (scanning QR codes attached on infected landmarks) or through the exposure of their avatars to virtual viruses in the simulation running in the virtual world. When this happens, they are informed through a message to their mobile device, at which point, they have to choose appropriate measures to protect themselves and the patients. This type of game play represents the so-called *astral-projection modality,* where virtual- and real-world players can interact with each other in any world, and activities are reflected in both worlds..

The other two game variants were deployed in a virtual world, allowing participants to play this game as a regular desktop video game. In other words, these two game variants were played in the *Dreaming modality*, and the only difference between these two versions was in the setting of the virtual-world space and in how players were able to interact with the game. One of the game variants uses the virtual world as a clickable 2D map (virtual 2D version), where participants have a top view of a Pacman board, on which players and viruses interact with each other, and participants can click on the items of the game in order to move around and play the game. The third game variant takes place in a 3D virtual world, where the virtual space configuration is similar to the physical space where the mobile version takes place, and players have a first-person view in a navigable virtual world.

The first game variant uses all the fAARS components, including two external systems as active subscribers. The other two game variants use the virtual-world actors, the game engine, and the same two external systems as active subscribers (similar to the HGO game). Let us now review the development process for the Outbreak game.

Actors. We created *Actor* instances for each of our student participants; some were real-world Actors (for the first game variant) and some were virtual-world Actors (for the other two variants). In addition, a number of *Actors* acting as *NPC*s (Non-player Characters) were deployed to represent the virtual-world viruses that infect the players during game-play.

Game Engine. The Outbreak game has many more rules than the HGO game that control the interactions of *Actors* with the QR codes representing questions, the QR codes outside the patients' rooms, and the collision with viruses. The *Event* types used to define the *ECA* rules of the game ware the *decodeQR* and *onCollision Events*.

Virtual World. This was an essential component for all three versions of this game. The *Astral Projection* and *Dreaming* game play modalities that supported the mobile and virtual 2D and 3D versions of this game use the *Virtual World* component. In the *Astral Projection* modality, the virtual world is used as an extension of the real world scenario, where all *Actor* instances could be infected by *NPC Actor* instances that represented viruses flying around in the game. In the *Dreaming* modality, it is in the virtual world where the interactions among all *Actor* and *NPC Actor* instances happened during game-play.

Event Subscribers. There were two Event Subscribers in this game that provided further downstream functionality. Both of these external components were used in all three versions of the game. The first was a repository of questions, which allowed players to increase their score if they correctly answered a question. The second component was a mobility controller that allowed to move *NPCs* around in the virtual world in a pseudo-randomly fashion. The algorithm used to update the location of the *NPCs* pseudo-randomly assigns a different location to an *NPC* every time it arrived at the target previously assigned.

5 Conclusions

The development of fAARS was motivated by our realization that, with the increased maturity and adoption of mobile and virtual-world platforms and momentum of the *"gamification"* paradigm in services (including education and health care), there is a great opportunity for trans-reality games. The fAARS platform supports a specific style of trans-reality games, grounded on a complex spatial metaphor and following the virtual treasure-hunt and RPG styles. The fAARS platform extends the state-of-the-art in pervasive location-aware gaming platforms in the following ways.

fAARS is the only platform that enables the deployment of location-aware games in a virtual world, as an extension of the physical indoors/outdoors playground. The virtual-world playground may "parallel" the real world (with all real-world and virtual-world coordinates mapped through a bi-directional function) or it may communicate a completely different metaphor to the game story (mapping only interesting locations across the worlds).

fAARS supports peer-to-peer interactions among players in the virtual world. Mimicking the functionality of NFC, a special-purpose software device in the virtual world recognizes player proximity. This event can be used by the game story to notify the players and give them opportunities to advance the game play.

Finally, fAARS enables a variety of combinations of real-world and virtual-world game-play. The two example games discussed in Section 4 demonstrate how fAARS can be used to develop real-world only games (like the platforms discussed in Section 2.2), or virtual-world only games, or true trans-reality games.

In the future, we plan to extend the platform to support more complex trans-reality games, with more interactions among the real-world and virtual-world players, including peer-to-peer interactions using Bluetooth and NFC. We also envision the inclusion of a wider spectrum of input sources, possibly sensors through which to enable adaptronic-style game play; this extension is quite straightforward given the fAARS event-based information exchange among its components. Finally, we plan to integrate an authoring environment, in order to simplify the process of game development.

References

1. Magerkurth, C., Cheok, A.D., Mandryk, R.L., and Nilsen, T. Pervasive games: Bringing computer entertainment back to the real world. *ACM Computers* i*n Entertainment*, *3,*3, (2005), Article 4A.
2. Special report: top 11 technologies of the decade, http://spectrum.ieee.org/static/special-report-top-11-technologies-of-the-decade (accessed February 2011)
3. Lindley, C. A. Trans-reality gaming. *In Proceedings of the 2nd annual International Workshop in Computer Game Design and Technology* (Liverpool, UK: Liverpool John Moores University, 2004), pp. 15-16.
4. Montola, M. Exploring the edge of the magic circle: Defining pervasive games, vol. 1966. *Citeseer* (2005), pp. 16-19.

5. Piekarski, W., and Thomas, B. Arquake: the outdoor augmented reality gaming system. *Commun. ACM 45* (January 2002), pp. 36-38.
6. Cheok, A. D., Goh, K. H., Liu, W., Farbiz, F., Fong, S. W., Teo, S. L., Li, Y., and Yang, X. Human pacman: a mobile, wide-area entertainment system based on physical, social, and ubiquitous computing. *Personal Ubiquitous Comput. 8* (May 2004), pp. 71-81.
7. Qui, T. C. T., Nguyen, T. H. D., Mallawaarachchi, A., Xu, K., Liu, W., Lee, S. P., Zhou, Z. Y., Teo, S. L., Teo, H. S., Thang, L. N., Li, Y., Cheok, A. D., and Kato, H. Magic land: live 3d human capture mixed reality interactive system. *In CHI '05 extended abstracts on Human factors in computing systems* (2005), CHI EA '05, ACM, pp. 1142-1143.
8. Benford, S., Crabtree, A., Flintham, M., Drozd, A., Anastasi, R., Paxton, M., Tandavanitj, N., Adams, M., and Row-Farr, J. Can you see me now? *ACM Trans. Comput.-Hum. Interact. 13* (March 2006), pp. 100-133.
9. Benford, S., Flintham, M., Drozd, A., Anastasi, R., Rowland, D., Tandavanitj, N., Adams, M., Row-Farr, J., Oldroyd, A. and Sutton, J. Uncle Roy All Around You: Implicating the City in a Location-Based Performance. In *Proc. Advances in Computer Entertainment (ACE 2004)*, ACM Press.
10. Flintham, M., J. Humble, N. Tandavanitj, M. Adams, J. Row Farr. I Like Frank: a mixed reality game for 3G phones. Submitted to *IEEE Computer Graphics And Applications*.
11. Herbst, I., Braun, A.-K., McCall, R., and Broll, W. Timewarp: interactive time travel with a mobile mixed reality game. *In Proceedings of the 10th international conference on Human computer interaction with mobile devices and services* (New York, NY, USA, 2008), MobileHCI '08, ACM, pp. 235-244.
12. O'Shea, P. M., Mitchell, R., Johnston, C., and Dede, C. J. Lessons learned about designing augmented realities. *IJGCMS 1, 1* (2009), pp. 1-15.
13. Gutierrez, L., Nikolaidis, I., Stroulia, E., Gouglas, S., Rockwell, G., Boechler, P., Carbonaro, M., and King, S. far-play: A framework to develop augmented/alternate reality games. *In PerCom Workshops (2011), IEEE*, pp. 531-536.
14. Suomela, R., Räsänen, E., Koivisto, A., and Mattila, J. Open-Source Game Development with the Multi-user Publishing Environment (MUPE) Application Platform. *In: ICEC Bd. 3166*, Springer, 2004, pp. 308--320.
15. Fetter, M., Etz, M., and Blechschmied, H. Mobile chase--towards a framework for location-based gaming. *In: GRAPP (AS/IE), INSTICC--Institute for Systems and Technologies of Information*, Control and Communication, 2007, pp. 98--105
16. Tutzschke, J.-P., and Zukunft, O. Frap: a framework for pervasive games. *In Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems* (New York, NY, USA, 2009), EICS '09, ACM, pp. 133-142.
17. Papakonstantinou, S., and Brujic-Okretic, V. Prototyping a context-aware framework for pervasive entertainment applications. *In Proceedings of the 2009 Conference in Games and Virtual Worlds for Serious Applications* (Washington, DC, USA, 2009), VS-GAMES '09, IEEE Computer Society, pp. 84-91.
18. Lindley, C. A. Game space design foundations for trans-reality games. *In Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology* (New York, NY, USA, 2005), ACE '05, ACM, pp. 397-404.
19. Michelson, B. Event-Driven Architecture Overview, Event-Driven SOA is just part of the EDA Story. *Patricia Seybold Group 2006*. Available from: http://www.omg.org/soa/Uploaded%20Docs/EDA/bda2-2-06cc.pdf (accessed March 2011).