Pervasive Games

Kim J. L. Nevelsteen
Immersive Networking, DSV, Stockholm University, Kista, Sweden

Synonyms
Ubiquitous games

Definitions
The first academic definition for a pervasive game (Montola 2012) is provided by Schneider and Kortuem (2001), defining a pervasive game as “a LARP [live action role-playing] game that is augmented with computing and communication technology in a way that combines the physical and digital space together.” In her pervasive discourse, Nieuwdorp (2007, original italics) highlights the ambiguity of the term “pervasive games” by listing the following meanings which she derived through a literature review (the reader is guided to her work for the sources of the review):

- A game that pervades the real world in an undefined manner, and thus blends with it
- A specific setting of the game world within the real world
- A game that blurs the boundaries between itself and the real world, which can influence the concept of the magic circle
- A game that is an overlay of the real world or where the world becomes a game board
- A game with a persistent presence in the real world, and thus available to the players at all times
- A game where the gameplay interacts with elements of the real world, thus challenging standard gameplay conventions
- A game where there is mutual interaction among players and elements in the real world
- A game that blends with everyday experiences

Nieuwdorp classifies these meanings into two perspectives: (1) “a technological one that focuses on computing technology as a tool to enable the game to come into being” (i.e., the first two meanings on the list above) and (2) “a cultural one that focuses on the game itself and, subsequently, on the way the game world can be related to the everyday world” (the last eight remaining meanings above).
Introduction

Although the origins of ubiquitous computing and pervasive computing differ (Nieuwdorp 2007), they are often used interchangeably (Nieuwdorp 2007; Montola 2012) and both are the basis for pervasive gaming (Nieuwdorp 2007). According to Montola (2005), “a pervasive game is a game that has one or more salient features that expand the contractual magic circle of play spatially, temporally, or socially,” i.e., “expand the boundaries of play” (Oppermann 2009).

Pervasive games have been linked to a number of similar genres, such as: ubiquitous games; augmented/mixed reality games; mobile games; alternate reality games; (enhanced) live action role play (E/LARP); affective gaming; virtual reality games; smart toys; location-based or location-aware games; adaptronic games; crossmedia games; augmented tabletop games (Nieuwdorp 2007); and exergames (Stanley et al. 2011).

Staging a Pervasive Game

Ståhl et al. (2007) have identified three temporal phases in staging a pervasive game: “pre-production,” “run-time,” and “post-production.” Because these phases concern games, these phases have also been referred to as “pre-game,” “in-game,” and “post-game” (Jonsson et al. 2007; Broll et al. 2006). The latter convention is used throughout this entry, reserving run-time to refer to when a game architecture is running and in-game to refer to when the game is running. In the pre-game phase, although resource demanding (Bell 2007), a game can potentially be adapted to each new staging (e.g., adapting to a new staging location (Oppermann 2009), i.e., supporting location adaptability). Adapting the game is done through reconfiguring the architecture and authoring content specific to each staging. Reconfiguration and authoring can continue into the in-game phase, provided it is supported by the architecture. In the post-game phase, an analysis of historic event data can be performed, players debriefed, and informed to the actual flow events (Stenros et al. 2007b). The results of a post-game analysis can inform further game design or stagings.

One of the driving factors why current game engines are ill suited for pervasive games is game mastering. Contrary to many video games, pervasive games do not necessarily run fully automatic. One or more persons, often referred to as game master(s) (GMs), can be assigned the responsibility of adjusting the game during its staging; an act which is referred to as game mastering (Jonsson and Waern 2008; Oppermann 2009; Montola et al. 2009) or orchestration (Thompson et al. 2003; Flintham et al. 2003a; Broll et al. 2006). A well-known role for a game master is that of a “puppet master”: in charge and “pulling strings,” all the while staying hidden behind the scenes (Jonsson and Waern 2008). Jonsson and Waern (2008) have argued that pervasive games benefit from being game mastered, e.g., allowing for: content to be actively authored to “fit the activities of the participants,” the altering of game events, adjusting of difficulty, providing dynamic gameplay, or the reincorporating of user response back into the game (Jonsson et al. 2007). Because pervasive games take place in the physical world, another responsibility of the game master is to keep players safe in the highly variable, possibly dangerous conditions of the physical world (Flintham et al. 2003a; Broll et al. 2006). A drawback of game mastering being that it can require a significant amount of human resources (Thompson et al. 2003; Flintham et al. 2003b). Jonsson and Waern (2008) have identified three needed functions, in order to successfully game master: (1) to be able to monitor the game, (2) make decisions about how the game should progress, and (3) have the ability to influence the game state.

Monitoring the Game

Players of pervasive games are mobile and out in the physical world. Two ways to monitor a player are: stationary hardware placed in the physical world (Stenros et al. 2007a; Jonsson and Waern 2008) or by giving players mobile devices to carry, interact, and communicate with (Jonsson and Waern 2008; Montola et al. 2009). The physical world affords seemingly infinite possibilities,
meaning players are always able to produce “soft” events, outside the awareness of the game architecture, but still in relation to the game (Jonsson and Waern 2008). To capture some of these soft events, players can be monitored through direct surveillance and accounts thereof registered in game architecture (Crabtree et al. 2004; Montola et al. 2009). To assist in picking up on soft events, players can also be tasked with self-reporting, in the form of diaries (Montola et al. 2009; Jonsson et al. 2007).

Perhaps not part of the game state, per say, but important in monitoring a game mastered game, is any meta-level information, e.g., game master notes instructing other game masters on the state of the game (Montola et al. 2009).

Support Decision Making
To support game master decision making in-game, the potentially massive amounts of event information during monitoring must be dealt with. Additional information to aid decision making includes semistatic information, such as player information (e.g., photo, contact details, emergency information, game relevant skills) or documented help on how to stage a specific event (Jonsson et al. 2007). Automation aids game masters in decision making but obviously reduces game mastering, leading increasingly to a fully automated experience. One option to reduce game master load, without increasing automation, is to provide support tools, e.g., in the form of specialized GM interfaces or log analysis tools (Montola et al. 2009; Broll et al. 2006; Dow et al. 2005). Support tools convert or condense event information into a human consumable format. Another option is to cast nonplayer characters (NPCs) (Souza e Silva and Sutko 2009) in to offload game master responsibilities (Jonsson and Waern 2008) achieving a more decentralized orchestration (Crabtree et al. 2004). Once game masters have made decisions, each decision must be actuated into the game.

Influencing the Game State
For a technology-sustained pervasive game, a common way to actuate change in a system, in run-time, is to directly alter the internal state of the game engine, i.e., alter variables in the game’s data space (provided it is possible to access it) (Jonsson et al. 2007; Jonsson and Waern 2008; Hansson et al. 2007; Broll et al. 2006). Depending on the architecture, not all modifications are possible in run-time, and in such a situation the system must be brought offline to make necessary modifications (Hansson et al. 2007). Manually manipulating variables in the data space can be cumbersome when authoring lots of content. Developers of a game can attempt to anticipate what part of the data space game masters need access to and build an appropriate GM interface to it.

Although important for relaying observed events in monitoring a game, communication also plays an important role in influencing the game state, e.g., by pushing information directly to the players (Jonsson and Waern 2008). Communication can be either diegetic or nondiegetic (Bergström 2011), with the nondiegetic channel being particularly important to communicate out of the context of the game, in case of emergencies (Jonsson and Waern 2008). (“The ‘diegesis’ of a story consists of whatever is true in that story. Diegetic elements are ‘in the story’; non-diegetic elements are not.” (Bergström 2011, original italics)) Communication channels can be uni- or bidirectional.

State of the Art
There is a class of pervasive games, which are “technology sustained,” relying on computer simulation to maintain game state and react to player activities; these games can be understood as “computer games interfacing with the physical world” (Montola et al. 2009, p.164). Technology-sustained pervasive games are contrary to “technology-supported” games, where not all game activities are supported by information technology (Montola et al. 2009), i.e., do not necessarily require a game engine. According to Broll et al. (2006), game engines for pervasive games do not differ entirely from computer games engines because “while the overall game is a mixed reality application combining the real
[physical] and the virtual, the game engine actually does not need to be aware of this fact.”

In a survey by Broll et al. (2006) important technologies for augmented-reality pervasive games have been summarized. Broll et al. (2006) mention the development of a pervasive game engine as “the logical next step,” but, unfortunately, do not discuss the details of what constitutes a pervasive game engine, e.g., in relation to concepts such as persistence, interoperability, game mastering, and communication.

A recent survey, by Kasapakis and Gavalas (2015), aims to classify pervasive games into age generations based on technologies used. The survey is limited to 18 pervasive games, e.g., not containing tabletop games, smart toys, and transreality games, even though being mentioned as subgenres in the articles’ related work. Kasapakis and Gavalas (2015) state that “the game engine organization model is largely dictated by the game scenario to be supported.” If a general-purpose pervasive games engine is to be created, commonalities between game technologies must be found, e.g., support for a virtual persistent world. Kasapakis and Gavalas (2015) continue that the “current technological status favours [sic] always-on connectivity, hence, centralized models”; this is not entirely correct since “always-on connectivity” relates to persistence, which can also be obtained through decentralized models.

In a recent publication by Kamarainen et al. (2014), the subject of cloud computing is discussed as a possible solution for pervasive and mobile computing, allowing the “end-user device to offload computation, storage, and the tasks of graphic rendering to the cloud.” Kamarainen et al. remark that latency is the “main challenge” for cloud gaming, with most interactive games requiring response times that only “local deployment scenarios” can deliver. As a solution, they “propose to use [a] hybrid and decentralized cloud computing infrastructure, which enables deploying game servers on hosts with close proximity to the mobile clients.” To exploit local resources, the Fun in Numbers (FiiN) platform features a distributed multitiered (i.e., four layered) large-scale architecture (Chatzigiannakis et al. 2011). The FiiN architecture supports more than one game engine, with each engine being the “local authority for each physical game site” (Akribopoulos et al. 2009). All game engines are coordinated by a centralized topmost layer.

The bottom layers of the FiiN architecture enable support for ad-hoc networks and Internet of Things (Gartner 2014). The problem with the FiiN architecture is that it is unclear exactly what types of pervasive games are supported. Pervasive games are defined in the FiiN publications as “games played in the physical space, indoors or outdoors, using mobile handheld devices, context-awareness, and in certain cases some degree of infrastructure and scripting.” Chatzigiannakis et al. (2011) make no distinction between technology-sustained or technology-supported pervasive games, even though technology-supported pervasive games do not necessarily require a game engine, i.e., the infrastructure to enable them is very different. Akribopoulos et al. (2009, original italics) state that FiiN targets “mainly games that involve multiple players, rapid physical activity, gesturing, … and less storytelling-based games,” which could account for why game master interfaces are not present in the architecture (Nevelsteen 2015).

**Technology-Sustained Pervasive Games**

Nevelsteen (2015) provides an extensive systematic review into pervasive games. Concentrating on technology-sustained pervasive games, a feature set is derived that describes characteristic features of a would-be pervasive games engine. These features can be considered a set of informal requirements from which a set of formal requirements can be drawn. Using the feature set, a virtual world engine was chosen as being in the same “product line” (Bass et al. 2013) as a would-be pervasive games engine, based on the shared trait of a persistence. The component feature set and the choice of a virtual engine as pervasive engine, by Nevelsteen (2015), are verified through the case study of the pervasive game called Codename: Heroes.
Component Feature Set
Having surveyed a total of 59 pervasive games/projects and 27 technologies, the following is a summary of the derived feature set for a would-be pervasive games engine (Nevelsteen 2015):

**Virtual game world with world persistence:** a spatiotemporal instance, with interacting virtual elements (at least one of which being the player); a game world that overlaps with both the virtual and the physical world; a world that continues to exist and develop internally even when there are no people interacting with it (persistence); and ubiquitous availability through a reliable architecture.

**Shared data space(s) with data persistence:** a common shared data space, with coordinated communication to it, and data persistence in the event of a shutdown or system failure, i.e., fault tolerant and recoverable.

**Heterogeneous devices and systems:** support for nonstandard input devices, comprised of sensors and actuators, that form an interface (Nieuwdorp 2005) between the player and the game; resolution of interoperability issues through a device abstraction layer; and the use of service-oriented architectures or the offering of such services.

**Context awareness:** context information, e.g., location, body orientation, available resources including network connectivity, proximity to surroundings or noise levels; context information is obtained through sensor enabled heterogeneous devices or service-oriented architectures; dealing with uncertainty in position localization or networking.

**Roles, groups, hierarchies, permissions:** various roles for player and nonplayer characters, organized in groups or hierarchies; different permissions or privileged information for the various roles or organizations, perhaps through an entirely different interface to the game.

**Current and historical game state:** including semistatic player info; a view of the current internal game state, e.g., through direct inspectable properties or through a specialized management interface; a historical perspective of the game state, e.g., through the logging of event data for post-game analysis; or any metalevel game information, such as game master documentation.

**Game master intervention:** the semiautomatic execution of the game through game master intervention in run-time, e.g., by directly manipulating the internal game state or through specialized interfaces, that potentially translate massive amounts collected game data into a human consumable form; game master intervention can be provide by a service-oriented architecture.

**Reconfiguration, authoring, and scripting in run-time:** in the pre-game phase (e.g., for location adaptability) that can be extended into the in-game phase; data-driven reconfiguration of software, hardware, and devices; dynamic story and content through content generation in-game; changing of the game rules through data-definition or run-time languages; autonomous agents; or the simulation of events for the Wizard of Oz (Dow et al. 2005) technique (see Section 2.3.8 of (Nevelsteen 2015)).

**Bidirectional diegetic and nondiegetic communication:** through various channels and/or interfaces.

Many of these features are quite generic (e.g., current and historical game state) and so are supported by engines in the domain of computer video games. Features more specific to pervasive games are heterogeneous devices and systems; context awareness; game master intervention; reconfiguration, authoring, and scripting in run-time; and bidirectional diegetic and nondiegetic communication, e.g., nondiegetic communication is required in a virtual world but not as pronounced as in a pervasive game where it is needed to cope with social expansion.

**Challenges and Open Issues**
The problems, caveats, and disadvantages highlighted through the case study of Codename: Heroes (Nevelsteen 2015) serve to highlight challenges and open issues for the creation of a would-be pervasive game engine. These challenges
include: (1) using distributed and decentralized architectures; (2) extending ubiquitous computing; (3) interoperability; and (4) creating game master interfaces and tools.

**Distributed and decentralized architectures:** Exemplified by Demeure et al. (2008), fully decentralized architectures exist, where the game state is not centrally controlled and only shared with other clients when opportune. A challenge exists pertaining to the extent that a decentralized architecture can be utilized for games. Issues arise as how to: maintain security, maintain a shared data space, and prevent cheating (Yahyavi and Kemme 2013); gather and persist data (e.g., for monitoring); or build dynamic user interfaces. To deal with the scalability issue, virtual world engines already exist that use a centralized distributed system of servers for load balancing (e.g., BigWorld (2011)), so utilizing such techniques for pervasive games seems evident.

**Extending ubiquitous computing:** Devices and systems have the potential to offer richer context information for context awareness, e.g., the incorporation of body metrics or social relations. Ubiquitous computing remains a challenge, with open issues: increased utilization of context awareness; reduction of soft events (e.g., in communication); focusing on technology that can be effectively pushed into the background (e.g., for ubiquity of access and diegetic communication); and obtaining ubiquitous persona and presence (Dionisio et al. 2013). The latter recognizing that a player’s identity is made up of the sum of their interactions with the game, e.g., crossmedia through different devices or interfaces. The amount of uncertainty in ubiquitous computing has been reduced considerably; early writings on pervasive games include much on mobile networking issues, which are solved in mainstream technologies today but some issues are still critical, e.g., losing connectivity by switching between WLAN and mobile networks. A partial solution could be that of delay-tolerate network communication, used in FinN (Akribopoulos et al. 2009), to obtain an eventually consistent game state in their distributed system.

**Interoperability:** In 2004, Greenhalgh et al. (2004) set out to interconnect heterogeneous devices with the EQUIP/ECT technologies. Broll et al. (2006) state interoperability in pervasive games to be a “well-known problem.” And, a number of years later, Branton et al. (2011) dedicate an entire publication to deal with the “important challenge” of interoperability through standardization. Many innovations, such as new languages or middleware, are sited by Branton et al. (2011) as partial solutions, but compatibility between web services was noted as “largely lacking.” Since some game engines and service-oriented architectures are already distributed systems, and they interact, then interoperability is an issue between heterogeneous distributed systems as well, i.e., similar to multi-cloud network communication (Singhal et al. 2013). Interoperability remains a challenge with the amount of heterogeneous devices and systems increasing and becoming more diverse (Gartner 2014).

**Game master interfaces and tools:** Because it is beneficial to build reusable game engines, it seems reasonable to infer that reusable game master interfaces and authoring tools should also exist (e.g., see (Broll et al. 2006)). Some game master tools have already been created (e.g., for mobile games (Paelke et al. 2008) and authoring tools for location-based games (Oppermann 2009)), but a more general reusable approach remains a challenge (Guerrero Corbi 2014; Benford et al. 2009). Open issues are: capturing soft events and entering them in the game state; reducing the potential overload of data into a human consumable format; creating interfaces and visualizations that are applicable to a wide variety of games; and generating interfaces and visualizations that cater to the activity of game mastering rather than just presenting information.

**Conclusion and Discussion**

Computer video games have existed for decades, with reusable game engines to drive them; the major incentive for employing a reusable game engine being reduced development time and cost (Lewis and Jacobson 2002; Bass et al. 2013). Technology-sustained pervasive games can be understood as computer games interfacing with
the physical world, so computer game engines can be used to stage a pervasive game (Nevelsteen 2015). Currently, there are no reusable game engines available for pervasive games, but herein a component feature set has been presented for a would-be pervasive games engine, including the challenges and open issues uncovered during the case study verifying the feature set.

According to Jonsson et al. (2007), pervasive games need a sensory system to monitor the physical world. Pervasive games are known to make use of “nonstandard input devices” (Nieuwdorp 2007), and with the rise of the Internet of Things (Gartner 2014), access to Internet of Things could potentially serve as such a sensory system. Considering the use of nonstandard input devices in pervasive games and the rise of the Internet of Things, how will this affect pervasive games remains an open question.

**Cross-References**

- **Physical, Virtual, and Game World Persistence**

**References**


Schneider, J., Kortuem, G.: How to host a pervasive game: supporting face-to-face interactions in live-action roleplaying. In: Designing Ubiquitous Computing Games Workshop at UbiComp (position paper), Atlanta, Georgia, USA, pp. 1–6 (2001)


