Chapter 1
Pervasive Games with Persistent Worlds

1.1 Introduction

According to Benford, Magerkurth, and Ljungstrand (2005), if traditional games have game elements in the physical world and computer games have game elements in the virtual, then "pervasive games extend the gaming experience out into the real [physical] world", with a representation of game elements in the physical, the virtual or a blend of both. Because the domain of pervasive games is broad and imprecise (Nieuwdorp, 2007), the focus of this book will be on pervasive games that satisfy the sub-domain criteria: pervasive games that *make use of virtual game elements*. In her definitive work, Nieuwdorp (2007, original italics) derives that pervasive games can be discussed from two perspectives, a technological one, "that focuses on computing technology *as a tool to enable the game to come into being*" or a cultural one, "that focuses on *the game itself*". This book is requirements and development-focused Design Science research (Johannesson & Perjons, 2014) examining pervasive games from a technological perspective, under the assumption that the technology utilized determines a set of pervasive games that can be supported from a cultural perspective e.g., position localization technologies enable location-based games.

According to Broll, Ohlenburg, Lindt, Herbst, and Braun (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". Paelke, Oppermann, and Reimann (2008) suggest a web server to stand as pervasive game engine. In the computer game industry, the use of a game engine to build games is common; the major incentive for employing a reusable game engine being reduced development time and cost (Lewis & Jacobson, 2002; Bass, Clements, & Kazman, 2013). If pervasive games are to reap the same benefits, then engines for pervasive games must be available (Paelke et al., 2008). But, current computer game

K.J.L. Nevelsteen, *A Survey of Characteristic Engine Features for Technology-Sustained Pervasive Games*, SpringerBriefs in Computer Science,
DOI 10.1007/978-3-319-17632-1_1

engines do not support pervasive games that move the game beyond the computer screen, out into the physical world, unbound by scheduled play times and possibly involving unknowing bystanders i.e., games that are expanded spatially, temporally and/or socially. Since the computer game industry is already rich with engines and game engines can be repurposed beyond their intended use (Lewis & Jacobson, 2002) (e.g., the use of the Quake (id Software, 1996) engine in the augmented reality game, called ARQuake (Oppermann, 2009)), this book investigates: (i) if a game engine can be repurposed to stage pervasive games; (ii) if features describing a would-be pervasive game engine can be identified; (iii) using those features, if an architecture be found in the same 'product line' (Bass et al., 2013) as an existing engine and if that architecture can be extended to stage pervasive games (iv) and, finally, if there any challenges and open issues that remain.

The approach to answering these questions is two fold: First, a survey of pervasive games is conducted, gathering technical details and features of various games, projects and technologies (see Chap. 2). Approaches, technologies and overall goals of each of the surveyed projects could differ greatly, including their aim to create either game-specific engines or more universal solutions. To stay within scope of the book, only those projects supporting the sub-domain criteria were considered. The gathered features are distilled into a component feature set that enables pervasive games and verified e.g., against the definitions of pervasive games and in comparison to related work. Second, a type of game engine is chosen (supporting as much of the feature set as possible) as candidate in the same product line as a would-be pervasive game engine. The architecture is extended to support the entire feature set and used to stage a pervasive game called Codename: Heroes (CN:H) as proof-of-concept (see Chap. 3). From the outset CN:H had the requirement of being a 'long term pervasive game', emphasizing 'stability' and 'scalability' by using a 'mature' (ISO, 2011) engine. Limited resources called for rapid development and using a game engine would allow designers to immediately implement the game play instead of implementation mechanics (Branton, Carver, & Ullmer, 2011; Suomela, Räsänen, Koivisto, & Mattila, 2004). Implementing CN:H serves to validate the architecture and give needed first-hand experience with the resulting architecture. Whereas features from the survey informed the architecture, the implementation of CN:H served to highlight those features of particular importance and identify any open issues.

Before presenting the survey in Chap. 2 and the case study in Chap. 3, the following sections contain: what a game engine is and its purpose; how game, physical and virtual worlds relate; a clarification pertaining to the ambiguities of persistence, in relation to the worlds and the game engine; a definition of pervasive games; and, an explanation of the three different temporal phases of staging a pervasive game.

1.2 Game Engine

When referring to a game's *architecture*, this can refer to both the software and hardware supporting the game. In order to divide-and-conquer complexity and promote reuse, software can be divided into components ('modules' (Bass et al., 2013)) that enable common features. A collection of components can be organized into a software architecture, which can be used as a game engine. Core functionality is often compiled into the game engine for performance reasons (Gregory, 2009). An engine can be data-driven,[1] with functionality (including the engine configuration) controlled through a scripting language (Gregory, 2009; Branton et al., 2011), alleviating the need for core engine programmers (Gregory, 2009; Greenhalgh, Izadi, Mathrick, Humble, & Taylor, 2004). The term *game engine* is sometimes used interchangeably with more specialized engines, such as a graphics engine, physics engine, virtual world engine, augmented reality[2] engine or middleware (Gregory, 2009). Gregory (2009) states the definition of a game engine to be "software that is extensible and can be used as the foundation for many different games without major modification". The more 'universal' (Gregory, 2009) an engine is, the more games can be supported (Hall & Novak, 2008; Lewis & Jacobson, 2002).

The benefits of using a game engine are reduced development time and cost. One way to reduce development cost, is to reduce the amount of code written, by reusing code e.g., if the cost of acquiring the engine far outweighs the development cost to implement the same feature set (Hall & Novak, 2008). Reduced development time is made possible by the engine, because designers are able to immediately start work on the actual game, while other components are still under development (Gregory, 2009; Lewis & Jacobson, 2002; Greenhalgh et al., 2004; Hall & Novak, 2008), satisfying both designers and technology experts (Paelke et al., 2008). If pervasive games are to reap the benefits of reduced development time and cost, then engines for pervasive games must be available.

1.3 Worlds

Various types of *worlds* are discussed in this book, qualified by the descriptors: game, physical or virtual. The abstract notion of a *game world*, according to Adams and Rollings (2006), is a reality created by pretending: "when the player enters the

[1] "The behavior of a game can be controlled, in whole or in part, by *data* provided by artists and designers rather than exclusively by *software* produced by programmers" (Gregory, 2009, original italics).

[2] A system combining the virtual and the physical enabling a real-time interactive three dimensional environment (Oppermann, 2009).

magic circle [the contractual magic circle of play, according to Montola, Stenros, and Waern (2009)] and pretends to be somewhere else, the game world is the place she pretends to be". The term *physical world* shall be used throughout the book, rather than 'real world' or 'reality', to avoid the discussion on what is real or not.[3] And lastly, to avoid entering the discourse on what a *virtual world* is (M. W. Bell, 2008), it is to be understood here as: minimally, a spatiotemporal instance with interacting virtual elements[4] in the instance, where one or more of the elements could be a representation of a player. Temporality in the instance allows for change.

If the game world the player pretends to be in, overlaps with the virtual or the physical, then at least some of the game elements must be present in the virtual or physical, respectively. A mixed representation is possible, but to exactly what extent the physical and virtual can overlap is beyond the scope of this book. If a game element exists in the physical, it may or may not have a virtual counterpart, and vice versa. If the game world overlaps with the physical, the gaming experience is extended out in the physical world, with the game world being part of a pervasive game.

1.4 Persistence

The term *persistence* is often used in conjunction with a virtual world (James et al., 2004), but is ambiguous. To resolve the ambiguity here, persistence is divided into three components: world, data and network. It is also possible to speak of a persistent game world, but aside from the three components, persisting a game world is from a done from a cultural perspective.

For a virtual world, *world persistence* refers to an environment that "continues to exist and develop internally even when there are no people interacting with it" (Bartle, 2003). The physical world is persistent and since pervasive games can blend with the physical, pervasive games are persistent worlds, sharing the trait of a persistence with virtual worlds (de Souza e Silva & Sutko, 2009a). In pervasive games, exact times of play (when the player is in the game world) can be expanded temporally, making times "remain uncertain, ambiguous, and hard to define" (Montola et al., 2009). A solution is to make the game world available at all times, as suggested by one of the definitions of pervasive games, "a game with a persistent presence in the real [physical] world, and thus available to the players at all times" (Nieuwdorp, 2007). Another solution is to simulate world persistence, by either scheduling play times around when the world is unavailable or by making relevant player/world data available at least when those player(s) reconnect to the game (Söderlund, 2009). To simulate a world developing internally, change can be generated relative to how much time has elapsed since the players connected.

[3] "Reality is that which, when you stop believing in it, doesn't go away" (Dick, 1978).

[4] Virtual elements are elements simulated by computers (Bartle, 2003; M. W. Bell, 2008).

If the game state (including player records, game objects,[5] or other data) maintained by the game engine, is said to be contained in its *data space* (Greenhalgh, Izadi, Rodden, & Benford, 2001), *data persistence* aims to ensure the preservation of the data space in the event of a shutdown or system failure i.e., ensuring 'fault tolerance' and 'recoverability' (ISO, 2011). The persistence requirement by James et al. (2004), stating that the world "retains records of player data indefinitely", corresponds to data persistence. If the data space is partially or entirely held in computer memory, data persistence can be achieved by writing out the data space to non-volatile storage e.g., in the form of a flat file or to a database. Contrary to persistence, when the state of the world is not critical, the game state can be left in volatile memory, providing neither fault tolerance or recoverability; the engine is restarted in the event of failure.

The last form persistence discussed is that of persistent connectivity or *network persistence* (Ståhl, Drozd, Greenhalgh, & Koivisto, 2006). But, considering pervasive games must contend with lots of uncertainty in connectivity (Oppermann, 2009) (see Sect. 2.3.4), network persistence is often not possible and something that is strived towards. In an attempt to take advantage of uncertainty, in the game called Treasure (M. Bell, 2007), the disconnected state was used as a game mechanic.

1.5 Pervasive Games

Although the origins of ubiquitous computing and pervasive computing differ (Nieuwdorp, 2007), they are often used interchangeably (Nieuwdorp, 2007; Montola, 2012) and are both the basis for pervasive gaming (Nieuwdorp, 2007). According to Montola (2012), "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". Montola (2012) goes on to recognize the definitive work by Nieuwdorp (2007), wherein the various meanings of pervasive games are summarized e.g., games that: depend on non-standard input devices,[6] are augmented by computers, blend the physical and virtual, or have a persistent presence in the physical world. It is precisely the definitions by Nieuwdorp (2007), that the feature set will be verified against in Sect. 2.5.

The term *pervasive game* has been associated with a number of subgenres: ubiquitous games, augmented/mixed reality games, alternate reality games, (enhanced) live action role play (E/LARP), virtual reality games, location-based games and

[5] "The collection of object types that make up a game is called the *game object model*. The game object model provides a real-time simulation of a heterogeneous collection of objects in the virtual game world" (Gregory, 2009, original italics).

[6] "A device is a combination of a hardware component and a software component, sending or receiving data. The software component may contain a driver, a library, or a software development kit" (Appelt, Ohlenburg, Greenhalgh, Oppermann, & Åkesson, 2008).

more (Nieuwdorp, 2007). Instead of focusing on a particular subgenre, the focus of this book is on the sub-domain of pervasive games, that satisfy the sub-domain criteria of: pervasive games that make use of virtual game elements. Such a minimal criteria means, games satisfying the sub-domain criteria might exist in several of the subgenres. The sub-domain was chosen as a minimum, to specify that computer simulation is used to support the game world of the pervasive game. The sub-domain is important to limit the scope of the survey in Chap. 2. Examples of games that fail to satisfy the sub-domain criteria are some ARGs or LARPs, which are perhaps technology-supported, but are not technology-sustained (Montola et al., 2009) i.e., have no need for a game engine.

1.6 Staging a Pervasive Game

Ståhl, Ohlenburg, Greenhalgh, and Nenonen (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, Waern, Montola, & Stenros, 2007; Broll et al., 2006). The latter convention is used throughout this book, 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 (M. 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, Montola, Waern, & Jonsson, 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 & Waern, 2008; Oppermann, 2009; Montola et al., 2009) or *orchestration* (Thompson, Weal, Michaelides, Cruickshank, & Roure, 2003; Flintham, Anastasi, et al., 2003; 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 & 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, Anastasi, et al., 2003; 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, Benford, et al., 2003). 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.

1.6.1 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, Montola, Waern, & Jonsson, 2007a; Jonsson & Waern, 2008) or by giving players mobile devices to carry, interact and communicate with (Jonsson & 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 & 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).

1.6.2 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 semi-static 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 (Bartle, 2003), 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 non-player characters (NPCs) (de Souza e Silva & Sutko, 2009b) in, to

offload game master responsibilities (Jonsson & 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.

1.6.3 Influencing the Game State

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 & Waern, 2008; Hansson, Åkesson, & Wallberg, 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 & Waern, 2008). Communication can be either *diegetic*[7] or non-diegetic (Bergström, 2011), with the non-diegetic channel being particularly important to communicate out of the context of the game, in case of emergencies (Jonsson & Waern, 2008). Communication channels can be uni- or bi-directional.

References

Adams, E., & Rollings, A. (2006). *Fundamentals of game design (game design and development series)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Appelt, W., Ohlenburg, J., Greenhalgh, C., Oppermann, L., & Åkesson, K.-P. (2008, April). *Deliverable D7.7: final software delivery of WP 7*.

Bartle, R. (2003). *Designing virtual worlds*. Indianapolis, Indiana, USA: New Riders Publishing.

Bass, L., Clements, P., & Kazman, R. (2013). *Software architecture in practice* (3rd). Addison-Wesley Professional.

Bell, M. (2007). *Guidelines and infrastructure for the design and implementation of highly adaptive, context-aware, mobile, peer-to-peer systems*. (Doctoral dissertation, University of Glasgow, Faculty of Information and Mathematical Sciences, Department of Computing Science).

Bell, M. W. (2008, July). Toward a definition of "virtual worlds". *Journal of Virtual Worlds Research, 1* (1).

[7] "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)

Benford, S., Magerkurth, C., & Ljungstrand, P. (2005). Bridging the physical and digital in pervasive gaming. *Communications of the ACM*, *48* (3), 54–57.

Bergström, K. (2011). Framing storytelling with games. In *Interactive storytelling* (pp. 170–181). Lecture Notes in Computer Science. Vancouver, Canada: Springer Berlin Heidelberg.

Branton, C., Carver, D., & Ullmer, B. (2011). Interoperability standards for pervasive games. In *Proceedings of the 1st international workshop on games and software engineering* (pp. 40–43). New York, NY, USA: ACM.

Broll, W., Ohlenburg, J., Lindt, I., Herbst, I., & Braun, A.-K. (2006, October). Meeting technology challenges of pervasive augmented reality games. In *Proceedings of 5th ACM SIGCOMM workshop on network and system support for games* (p. 28). Singapore: ACM.

Crabtree, A., Benford, S., Rodden, T., Greenhalgh, C., Flintham, M., Anastasi, R., . . . Steed, A. (2004). Orchestrating a mixed reality game 'on the ground'. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 391–398). CHI '04. Vienna, Austria: ACM.

de Souza e Silva, A., & Sutko, D. M. (2009a). Digital cityscapes. (Chap. Merging Digital and Urban Playspaces: An introduction to the Field, pp. 1–20). USA: Peter Lang.

de Souza e Silva, A., & Sutko, D. M. (Eds.). (2009b). *Digital cityscapes*. USA: Peter Lang.

Dick, P. K. (1978). *How to build a universe that doesn't fall apart two days later*. Retrieved from http://deoxy.org/pkd_how2build.htm

Dow, S., Lee, J., Oezbek, C., MacIntyre, B., Bolter, J. D., & Gandy, M. (2005). Wizard of oz interfaces for mixed reality applications. In *CHI '05 extended abstracts on human factors in computing systems* (pp. 1339–1342). New York, NY, USA: ACM.

Flintham, M., Anastasi, R., Benford, S., Drozd, A., Mathrick, J., Rowland, D., . . . Sutton, J. (2003, November). Uncle roy all around you: mixing games and theatre on the city streets. In *Level up conference proceedings*. University of Utrecht: DiGRA.

Flintham, M., Benford, S., Anastasi, R., Hemmings, T., Crabtree, A., Greenhalgh, C., . . . Row-Farr, J. (2003, April). Where on-line meets on the streets: experiences with mobile mixed reality games. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 569–576). CHI '03. New York, NY, USA: ACM.

Greenhalgh, C., Izadi, S., Mathrick, J., Humble, J., & Taylor, I. (2004, September). ECT: a toolkit to support rapid construction of ubicomp environments. In *Proceedings of workshop on system support for ubiquitous computing*. UbiSys. Nottingham, UK: Springer Verlag.

Greenhalgh, C., Izadi, S., Rodden, T., & Benford, S. (2001). *The EQUIP platform: bringing together physical and virtual worlds*.

Gregory, J. (2009, July). *Game engine architecture* (J. Lander & M. Whiting, Eds.). Wellesley, Massachusetts: A K Peters.

Hall, R., & Novak, J. (2008, April). *Game development essentials: online game development*. Clifton Park, NY, USA: Delmar Cengage Learning.

Hansson, P., Åkesson, K.-P., & Wallberg, A. (2007, February). *Deliverable D11.9: second generation core platform*.

id Software. (1996). Quake.

ISO. (2011). *ISO/IEC 25010:2011 systems and software engineering – systems and software quality requirements and evaluation (SQuaRE) – system and software quality models*.

James, D., Walton, G., Mills, G., Welch, J., Valadares, J., Estanislao, J., & DeBenedictis, S. (2004). *2004 persistent worlds whitepaper*.

Johannesson, P., & Perjons, E. (2014). *An introduction to design science*. Springer International Publishing Switzerland.

Jonsson, S., & Waern, A. (2008). Art of game-mastering pervasive games, the. In *Proceedings of the 2008 international conference on advances in computer entertainment technology* (pp. 224–231). ACE '08. New York, NY, USA: ACM.

Jonsson, S., Waern, A., Montola, M., & Stenros, J. (2007, June). Game mastering a pervasive larp. experiences from Momentum. In *Proceedings of the 4th international symposium on pervasive gaming applications* (pp. 31–39). Salzburg, Austria: PerGames.

Lewis, M., & Jacobson, J. (2002, January). Game engines in scientific research. *Communications of the ACM*, *45* (1), 27–31.

Montola, M. (2012, September). *On the edge of the magic circle: understanding pervasive games and role-playing*. (Doctoral dissertation, School of Information Sciences).

Montola, M., Stenros, J., & Waern, A. (2009). *Pervasive games. theory and design. experiences on the boundary between life and play*. Burlington, MA, USA: Morgan Kaufmann Publishers.

Nieuwdorp, E. (2007, April). The pervasive discourse: an analysis. *Computers in Entertainment (CIE)*, *5*.

Oppermann, L. (2009, April). *Facilitating the development of location-based experiences*. (Doctoral dissertation, The University of Nottingham).

Paelke, V., Oppermann, L., & Reimann, C. (2008). Mobile location-based gaming. In *Map-based mobile services- design, interaction and usability* (Chap. 15, pp. 310–334). Springer Berlin Heidelberg.

Söderlund, T. (2009). Digital cityscapes. (Chap. Proximity Gaming: New Forms of Wireless Networking Gaming, pp. 217–250). USA: Peter Lang.

Ståhl, O., Drozd, A., Greenhalgh, C., & Koivisto, A. (2006, August). *Deliverable D6.7: second phase release of the IPerG platforms*.

Ståhl, O., Ohlenburg, J., Greenhalgh, C., & Nenonen, V. (2007, August). *Deliverable D6.8: final release of the IPerG platforms*.

Stenros, J., Montola, M., Waern, A., & Jonsson, S. (2007a, May). *Deliverable D11.8 appendix c: momentum evaluation report*.

Stenros, J., Montola, M., Waern, A., & Jonsson, S. (2007b). Play it for real: sustained seamless life/game merger in momentum. In *Situated play* (pp. 121–129). Tokyo, Japan: DiGRA.

Suomela, R., Räsänen, E., Koivisto, A., & Mattila, J. (2004). Open-source game development with the multi-user publishing environment (MUPE) application platform. In *Entertainment computing – ICEC 2004* (pp. 308–320). Eindhoven, The Netherlands: Springer Berlin Heidelberg.

Thompson, M. K., Weal, M. J., Michaelides, D. T., Cruickshank, D. G., & Roure, D. C. D. (2003). *MUD slinging: virtual orchestration of physical interactions*. ECSTRIAM03-007.