Simulation Gaming OnlineFirst, published on July 24, 2009 as doi:10.1177/1046878109340295

Simulation & Gaming
XX(X) xx–xx
© The Author(s) 2009
Reprints and permission: http://www.sagepub.com/journalsPermissions.nav
DOI: 10.1177/1046878109340295
http://sg.sagepub.com

Challenges of Online Game Development: A Review

Graham Morgan[1]

Abstract

The focus of this article is to determine how the engineering practices common in online game development may be approached differently to promote more diverse online gaming scenarios. The technical difficulties in providing online gaming are not trivial, requiring substantially larger budgets compared with their non-online counterparts: Commercial failure of an online game could be costly. Therefore, when considering alternate engineering approaches, those that appear tried and tested in other domains, and hence may be lower-cost solutions, are considered.

Keywords

commercial success, content management, content sharing, CORBA, cost, engineering approaches, evolution, game development, game scenarios, interaction, interest management, interoperability, middleware, MMO, multiplayer gaming, networking, online gaming, performance, programming, propriety play, protocols, rules, scalability, service agreement, standardization, standards, technical difficulties

This article is structured as follows. First, background information relating to the online gaming industry is described followed by a number of technical challenges faced by online game developers. A number of research avenues are then suggested, which may overcome these challenges. Finally, conclusions are presented.

In the opening section, we emphasize the significant cost of bringing an online game to market. An argument is suggested that the high cost to market entry is due

[1] School of Computing Science, Newcastle University, Newcastle Upon Tyne, UK

Corresponding Author:
Graham Morgan, School of Computing Science, Newcastle University, Newcastle Upon Tyne, NE1 7RU, UK
Email: graham.morgan@ncl.ac.uk

to not only the very difficult technical challenges associated to online game development but also the manner with which existing engineering approaches are employed to overcome such challenges.

Apart from pointing out technical difficulties, this article presents an optimistic view that there already exists significant work that could aid in online game development. Such work is not always considered by game developers and is often overlooked by researchers in game development. This oversight is primarily caused as such work is achieved in a different application domain (e.g., eCommerce) and is yet to make a significant impact in online game development.

The technical challenges covered in detail in this article are as follows:

- *Propriety play:* Competitive commercial activity makes it impossible for players to move their gaming content, possibly created at some expense, between online games created by different vendors.
- *Longevity:* Current approaches to maintaining content in online worlds are extremely costly and afford little scope for exciting change once content is introduced.
- *Describing play:* Existing ad hoc approaches to describing interaction in a game make it difficult to model intricate interaction to afford a wide variety of game play.

The counterpart to these challenges, the potential for optimism if you will, is the wealth of already accomplished work described, within this article, in the following sections:

- *Standardization quest:* Distributed application development (especially Internet applications) have benefited from standardization for many years.
- *Safe and useful content evolution:* Dynamically updating data and behavior of Internet-based computer systems is not too dissimilar to virtual-world content management and is already a well-explored discipline.
- *Enabling rich interaction:* Describing services (allowing automatic searching of them) and the monitoring of interaction between multiple computer-based participants has long occupied the eCommerce developer.

This article does not prescribe finished solutions to the most demanding technical challenges of online gaming. Nor are the accomplishments of online game developers and researchers belittled; online games are fully functioning, highly profitable, systems. Indeed, the advances in game technology (not just online gaming) have been quite substantial over the years. The article (also published in this journal series) by Prakash et al. (in press) describes many of these advancements quite eloquently and highlights just how technically demanding a subject game development is.

As a researcher, one must always consider opportunities to diversify approaches to discover novel solutions. To prompt such diversification, this article seeks to encourage avenues of research rarely considered in the games industry in an attempt to circumvent some of the technical challenges faced by the online games industry.

Multiplayer Gaming

A number of technological factors influence game play possibilities for online gamers. This goes beyond the limitations of play imposed by the user interface itself to include the networking limitations of the Internet. The enabling technologies responsible for managing game state across multiple-player consoles dictates what a player can and cannot do in an online virtual world. From the network protocols through to the storage and representation of game artifacts on remote servers, the approach to implementation taken by a game developer will inevitably influence the rules of a game and dictate the type of games possible.

Diverse Skill Sets

Considering that some online games may take many teams of developers covering many skill sets (e.g., networking, graphics, databases, clustered computing) a number of years to produce, realizing the ramifications on game rules even the smallest change to implementation may cause is difficult to judge. In turn, this makes judging what may be possible in a multiplayer game a difficult task at the design stage given the need to realize all implementation issues throughout the development, deployment, and maintenance of an online game.

The diverse computational skills that combine to make online games result in complicated pieces of software. Jim Waldo (2008), who has worked in distributed systems research for a number of years with great success, points out that online games present quite different challenges to other distributed application domains (e.g., eCommerce). However, the basic problems remain of gaining scalability and ensuring correctness of execution.

In an effort to ensure correctness of execution (and stave off virtual-world failure), tried and tested approaches to virtual-world creation are repeated time and again. This has resulted in many virtual worlds appearing very similar to each other. This, in turn, has resulted in similar game play scenarios across many different virtual worlds with minimal in-world improvements or additions to excite the player. Advances in technology appear to be minimal as virtual-world numbers increase, and what is on offer to the player seldom changes or varies across vendors.

Billion Dollar Industry

Online games are here, and they are a multibillion dollar industry. Money is made, and such commercial ventures have proved highly successful. Therefore, one may wonder why we hint that online gaming is, in technological terms, in its infancy. The alternative should be considered: Will enabling technologies emerge to ease the development of online games and allow more interactive, complex gaming arenas? The cost of bringing an MMO (Massively Multiplayer Online—industry neglects to include "game" in the acronym) game to market may be in excess of $10 million (Carpenter, 2003), with some placing the figure closer to $50 million (Zenke, 2008). In addition, once an online game is up and running, the maintenance costs may require a total investment, including start-up, of close to $500 million to contemplate competing as a market leader (Zenke, 2008). These figures seem substantial considering that many successful Internet applications are brought to market for much less. Will technology specifically designed for online gaming lower the entry level for such enterprises?

Propriety Play

Online gaming spans many different genres of play. From the ability to gamble online to the ability to participate in a first-person shooter game, players are provided with a number of gaming choices. Different gaming environments are provided via different online services. For example, some vendors may specialize in Web sites that provide poker games, while other vendors may specialize in providing services that host first-person shooter environments. These are quite separate businesses, and one would not expect to come across gamers playing Texas Hold'em poker within an online server supporting the popular first-person shooter MEDAL OF HONOR (EA, 2008). Not only is the game play distinctly different, but the requirements of these two gaming genres have resulted in quite different technological approaches to engineer.

Different Engineering Approaches

The provision of specialized gaming services over the Internet via a multitude of different vendors using different engineering approaches is expected, both by vendors and players alike. In fact, a question one may pose when considering an integrated engineering approach to diverse gaming environments such as poker and first-person shooters is "Why would anyone ever want to play poker in the middle of a fire fight?" This question may be considered for all different game genres that do not lie comfortably together. However, by considering a negative answer to this question, we encourage developers to consider the engineering of solutions whose only commonality is defined by the standard Internet protocols (they must at least work over the Internet) (Postal, 1981).

Minimal Standardization

Without a concerted effort toward standardization, technologies built on top of such protocols for the support of gaming environments will be propriety in nature. Furthermore, such technologies will tend to be created with particular gaming genres in mind. This is a hindrance to code reuse, one of the fundamental requirements in software engineering practices.

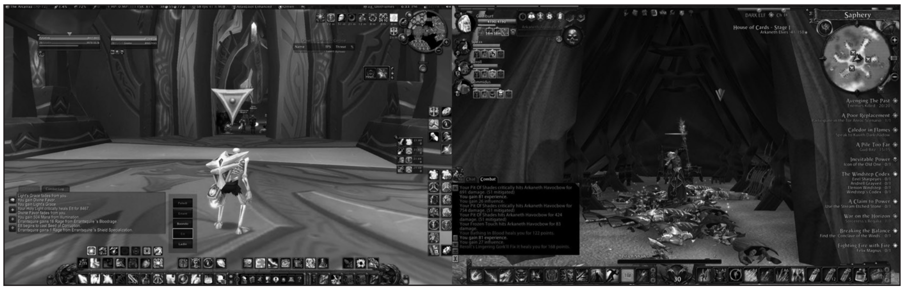

Figure 1. Similar representations but two different games

The greatest drawback to interoperability, and a truly interoperable gaming platform, is the inability of the games industry to apply standardization at the content level. Online game play and content are provided on a per-platform basis. Each platform is propriety to a particular online gaming vendor. The lack of existing work, academically and within industry, in this area gives no indication that this will change in the near future.

Inability to Share Content

At present, two different persistent virtual worlds (provided by different vendors) cannot share content. No standard exists to allow the representation of a game artifact, rule, or context to be described in the same manner for all online environments. For example, gaming scenarios and content developed for WORLD OF WARCRAFT (WoW, 2008) is limited to the WORLD OF WARCRAFT domain. Even similar domains, such as LORD OF THE RINGS (LoTR, 2008) and WORLD OF WARCRAFT, that share similar gaming interfaces, storyline progressions, and character development have no concept of sharing content.

Figure 1 presents screenshots from games where similarities are quite evident (WORLD OF WARCRAFT and WARHAMMER ONLINE). These similarities are to be expected as the graphical representation of the artifacts are developed using similar graphical tools and are saved in a format which, if not the same, could be transferred from one format to another using the appropriate tools (e.g., 3D Studio Max to DirectX).

More Than Graphical Representation

To ensure appropriate interoperability at the content level, we need to go beyond graphical format issues; the way artifacts are stored in persistent storage and the attributes associated with them need to be standardized in some manner as well as the manner with which players interact with them. This was viewed as a problem by the U.S. Department of Defense in the mid-1990s as efforts were made to ensure that simulation projects that were funded by the military could actually integrate and work successfully together (and so bring greater value for money). This effort of standardization of content across different platforms was proposed via the high-level architecture (HLA; Miller, 1996).

The HLA does provide a platform for interoperability of virtual worlds given heterogeneous deployment environments but has not found favor within the commercial games industry. The HLA has been limited to military simulations as opposed to general purpose gaming. However, the work carried out by Fong (2006) describes how off-the-shelf components can be used for low-cost military simulations, indicating that 10 years later, this was still a target for military-oriented simulations. Therefore, an assumption may be drawn that such interoperability is still difficult to achieve even with concerted effort.

Promoting Interoperability

Bringing standardization to middleware and content provision will allow interoperability of game play for players across different gaming platforms and gaming scenarios. At the moment, technology creation is viewed as a competitive strategy for game developers, forcing players to make a choice (and pay) for different gaming scenarios, and a competitive edge, providing something that is a little different or a little better than a competitor. However, the cost of developing MMOs has reached millions of dollars and the expense of maintaining such worlds have also reached millions of dollars; this has resulted in a high entrance level to market, actually hindering rather than helping the industry expand.

Longevity

Many popular gaming scenarios provide virtual worlds that are persistent in nature (e.g., WoW, 2008). Players return to participate in persistent worlds many times over a period of weeks, months, or even years. Persistent virtual worlds allow participants to enter a virtual world that provides a degree of continuity; artifacts may be created and persist over periods of time, and the results of events on artifacts may persist. For example, a participant may purchase a virtual car, drive the car to the end of a virtual road, return some days, months, or even years later and retrieve the car. Of course, someone else may have procured the car and driven it elsewhere in the meantime, but the continuity provided by persistence of artifacts is a factor that aids in classifying these virtual worlds.

Maintaining Interest

Player interest must be maintained over prolonged periods of time to ensure the financial success of an MMO: The longer players participate in an MMO, the more revenue such players will supply as they contribute their financial subscriptions and/or participate in virtual-world financial transactions. Therefore, an important factor in the design of any MMO is the requirement to continue providing new and challenging scenarios to encourage player participation. This can be achieved by periodically introducing new content (e.g., artifacts, rules, stories, areas) and ensuring that all content exhibits a degree of persistence to provide a heightened sense of continuing community. For the purposes of this article, the interesting questions relating to the quality of game content is not considered. The interested reader is directed to other literature for this interesting subject (e.g., Ang, 2006).

Evolution of Content

Apart from a recent discussion related to scripting in gaming (White, Koch, Gehrke, & Demers, 2008), one may be surprised to learn that little has been achieved academically to ease content maintenance and creation in persistent online worlds. This is surprising because of the importance vendors place on maintaining a constantly evolving virtual world to retain players. Evolutionary change is afforded only at the coding level via manual updates. This approach has resulted in the management of evolutionary change in an ad hoc manner and severely limits the ability to introduce far-reaching change while ensuring the correctness of a virtual world.

To illustrate how challenging a problem may be, consider the following example.

> In a virtual world that already allows users to navigate ships between ports, we wish to evolve an economic market by introducing "trade" and "cargo." Once introduced, users will be able to trade between ports via ships carrying cargo. This enhancement to the gaming scenario requires the modification to the artifact ship to enable the carrying of cargo. In addition, the new concept of trade will require modification to the rules governing the virtual world itself. Ports will assume the role of trade hubs and must be enhanced to recognize their role in trading.

In this example, it is not sufficient to just add content, but existing content (ships, ports) must also be altered to enhance them with the ability to participate in trade. This requires updates in the persistent data store (e.g., amount of cargo that a ship can carry) and updates in the application logic to enhance functionality (e.g., unload/load cargo). Furthermore, other artifacts not mentioned in our example must be designated as cargo. This in itself will require updates to other artifacts in the persistent data store (e.g., weight, size, and owner) and additions in the application logic (e.g., in transit, set owner, and change value). Finally, the concept of trade itself is quite fundamental and not easily captured within one single artifact, requiring recognition in the rules governing a virtual world (e.g., supply, wealth, and exchange).

A Commercial Problem

Commercial activities have shown the problem of managing change in persistent virtual worlds to be acute (CNet, 2006). Vendors are restricted to manual updates by their own developers or by players. This is primarily because of an inability of the technology to be flexible enough to self-manage content change. Vendors may have good reasons to manage content for the purposes of coherent story lines and directing the overall look and feel of a gaming scenario. However, this is a burdensome task when millions of items exist. Therefore, an alternative approach has arisen where players are encouraged to create such content, albeit at the expense of a vendor's ability to direct gaming scenarios (Linden, 2008).

When vendors manage content, the use of client-side updates coupled with additions at the server side is common as can be seen with the introduction of THE TRIALS OF OBI-WAN introduced by Lucas Arts (Lucas Arts, 2008) to enhance their STAR WARS GALAXIES MMO. Updates to client's software are an additional revenue stream for a vendor. Such updates are achieved by the vendor releasing "expansion packs" (software updates) that the player must purchase to participate in new gaming scenarios. To ensure that existing players may continue to participate without "expansion packs," the vendor isolates new scenarios from existing content. This is achieved by adding a new area to a virtual world. In reality, existing content is not evolved but increased in the form of additional areas.

SECOND LIFE (Linden, 2008), by Linden Lab, allows player-created content with a financial revenue model based on real estate and trading: The main type of revenue for Linden Lab relates to the purchase of land and paying of ground rent. Such content may then be traded between players. No client-side updates are required to access new content. A scripting language allows items of a virtual world to be instilled with behavior, allowing players to provide their own gaming scenarios. This approach provides SECOND LIFE players with the most powerful content creation tool available today for persistent virtual worlds with players providing a wealth of content.

Content Management Problems

Existing approaches to vendor- and player-derived content evolution can't realize our trading example, as existing content cannot be changed appropriately to accommodate new content. In SECOND LIFE, propagation of change from one artifact to another is limited and inhibited between artifacts belonging to different owners. Even using such an inhibitive approach, SECOND LIFE has been plagued by problems (failure of simulation because of erroneous scripts) (Linden, 2007). The more controlled approach used in vendor-driven content change has faired better in terms of virtual-world correctness (but failures still happen). However, this safety has come at the expense of limiting existing content updates to simple bug fixes and only allowing new content distinct from existing content. Fundamentally, all existing approaches severely limit content evolution in favor of safety, and the programming burden is immense.

Describing Play

The need to provide some mechanism to describe interaction in any gaming environment is a necessity, as without such a mechanism, no way exists to describe a game.

Describing a game is important, if one wishes to reason about the possible behavior in a game and if such behavior would result in undesirable gaming scenarios. For example, a virtual world in which players may assume much greater abilities than others will ultimately lead to an unbalanced gaming scenario.

Interest and Influence Management

In a real game, different interactions indicate different types of events, and the previous occurrence of events cumulatively contributes to the current game state. The current game state then influences player participation, which manifests itself as further events witnessed by players. One way of describing the abstract notion of game progression is via the identification of interest expressed and influence exerted by players, and the gaming environment itself, during a game. In practice, the behavior that may be exhibited is quite extensive given the limited expressiveness, as shown in Noy, Raban, and Ravid (2006), but for the purposes of this article, we consider such expressiveness difficult to describe.

Interest management is the term used to describe how influence exerted and interest expressed by participants in a virtual world translates to participant interaction. For example, consider a plane carrying food parcels. A plane may influence a food parcel by “carrying it.” A food parcel may influence a plane by “making it heavier.” A food parcel “dropped” by a plane (plane influencing food parcel) may provide nutrition to individuals (food parcel influencing individuals).

Determining Influence

Identifying when influence occurs appears straightforward (e.g., parcel in plane makes plane heavier) and is based on where items are in a virtual world. However, this may not always be the case as the relationships between the different items inhabiting a virtual world may lead to behavior not necessarily considered by a developer.

In the parcel example, there are transitive relationships to consider (heavier plane indicates increased fuel consumption; therefore, parcel requires more fuel for transit). Consideration of the laws of physics (acceleration) must be taken into account, which may influence the tactics of pilots and air traffic controllers (the height flown at, the use of jet streams, acceleration strategies, etc.). This may be straightforward to describe in isolation, but in complex scenarios the eventual behavior may be difficult or even impossible to predict.

Language for Interaction

Describing the manner of interaction between participants requires a language capable of expressing a variety of techniques. Although some languages have been proposed (e.g., Powel, Mellon, Watson, & Tarbox, 1996), they tend to be limited in their expressiveness. Such limitation is witnessed in many online gaming genres and reflects (like the technology used to implement them) specific interaction patterns.

In this respect, a card game implements a particular style of interaction, whereas a first-person shooter will implement another style of interaction.

Geography as a Basis for Determining Interaction

In commercial products, interest management schemes identify potential player interaction based on geographic location of participants in a virtual world. Popular games in the MMO genre all employ clustered-server solutions to achieve scalability while managing consistency. The techniques used to implement their interest management solutions in a server cluster is not described in detail in a published article for general viewing (which is to be expected for a commercial enterprise in a competitive market). However, an article (Kushner, 2005) describing EVERQUEST's approach in general terms does exist: a mixture of regions and "duplicate worlds" (duplicate worlds are sometimes called *shards*) with each duplicate world supporting approximately 2,000 to 3,000 players with each world divided into regions based on the geography of the virtual world.

In EVERQUEST, a duplicate world is itself supported by a cluster of servers, with regions used to aid in allocating the processing requests that originated from player actions among such servers as and when required. Because of the similarities in game play and the existence of duplicate worlds; one may assume that all other commercial MMOs approaches to implementation of interest management are similar, conceptually, to that of EVERQUEST.

Duplicate worlds and geographically influenced regionalization present a three-step approach to reducing the identification of interest and influence to a manageable size: (a) players do not interact across different duplicate worlds; (b) players do not interact across different regions; and (c) players interact intricately with other players they specifically target (e.g., click on with mouse). This approach provides two distinct forms of interaction: (a) a general, viewing type style, where players can see the actions of others in their region (assuming appropriate line of sight) and (b) an intricate manner where players directly interact with each other in a user-directed way.

This author, together with his research team (Morgan, Lu, & Storey, 2005), has shown that it is possible to allow dynamic configuration of the virtual space to model interaction while maintaining scalability. This is compared with the regionalization approach taken in industry. Dynamic approaches do allow a degree of interaction more accurate than the three-step approach, allowing different virtual-world inhabitants to specify their own areas of influence without regard for static regions. Industry thought that this approach may not be scalable as earlier academic work found it difficult to make such a solution scalable beyond a few users. However, with judicial use of enabling technologies at the server-side, player numbers can be supported on the scale required by commercial solutions (Lu, Parkin, & Morgan, 2006).

Synchronization and Interest Management

Once an assumption is made that interest management affords greater scalability, a developer must consider how messages may be synchronized within an interest management scheme to ensure that players are presented with a mutually consistent view of a virtual world. Messages sent without consideration of synchronization may be received in arbitrary order, or worse still, not arrive at all (if communication channels are not reliable). This would ultimately result in players witnessing the virtual world in different ways, hindering game play.

Approaches to message synchronization for virtual worlds using interest management techniques for scalability find their origins in the parallel and simulation community. Roccetti, Ferretti, and Palazzi (2008) provide a convenient discussion of the state of the art together with some interesting comparisons made between the techniques. In addition, work by Minson and Theodoropoulos (2008) indicates that such approaches may be suitable in first-person shooter-type environments. However, the consistency protocols that afford the best synchronization (developed for fault-tolerant computing) have had little success transferring to online gaming. One reason could be the difficulty such protocols had of scaling beyond 10 or 20 players when employed in virtual worlds in the early 1990s. This is most evident when considering the utilization of the ISIS group communication toolkit (Birman, 1986) to synchronize participant interaction within the DIVE system (Carlsson & Hagsand, 1993). Less than 20 participants could be supported before the system slowed to unacceptable levels.

Scalability and Interest Management Confusion

What has been witnessed in commercial solutions to describing game play (due in part to academic work) is confusion between the use of interest management to model player interaction and the exploitation of interest management to engineer a scalable solution; interest management appears to define the barriers to interaction as opposed to promoting interaction. Once interest management is exploited to provide scalability, there must be additional mechanisms implemented to promote synchronization. This provides a competing balance between synchronization on one hand and scalability on the other. However, one must not look too dimly on the outcome of this research, as the inability to provide a scalable solution would inhibit commercial activity (as to turn players away from a game would not make good business sense).

Standardization Quest

In recent years, middleware for game developers has appeared that may indicate some movement to standardization.

Console Vendors Promote Standards

The recent consoles (Xbox 360, Wii, and PS3) are designed to make full use of network capabilities, being both LAN and Internet ready. Microsoft provides a comprehensive development environment via its XNA toolset (XNA, 2008), which is based on Microsoft's own .NET framework. This leverages a mature middleware standard tailored for the game developer. Targeted at all developers (from professional to amateur), this middleware has seen success (as witnessed in its XNA Creators Club) and encourages a number of games to be released on Xbox Live and PC platforms. Xbox Live is the media delivery system associated with Microsoft's Xbox gaming consoles (as well as limited PC capabilities). Sony has its own competitor to Xbox Live called Playstation Network, designed for use with its PS3 console. This offers similar services to the end user as Xbox Live. However, the development environment is not as forthcoming compared with XNA; typically because Microsoft is primarily a software-based company as opposed to Sony's hardware focus (one would expect Microsoft to be stronger in this area).

Server-Side Standards for MMOs

Development of scalable server-side solutions for gaming environments have been tackled via the provision of middleware. An important advancement is the attempt IBM made to use an existing, well-known, middleware standard in the game development process (Shaikh, Sahu, Rosu, Shea, & Saha, 2006). During the excitement of grid computing, IBM attempted to demonstrate the usefulness of their Grid Technology and Web Services for the support of online game development (Sharp, 2004). This is an example of a concerted effort, at some financial cost, to encourage the game development community to consider a standardized approach to online game development. An interesting aspect of this work is the recognition that content itself needs to be standardized with an identification made by IBM of the business value chain involved in producing an MMO (with different vendors producing different technologies). Figure 2 highlights this approach. Another notable subscriber to this approach is Sun Microsystems, which is creating a server-side clustering solution (Project Darkstar) based on its well-known Java technologies (Sun, 2008b). Although Sun's approach is not focused on standardized content, it does promote the use of existing technologies for achieving scalable server-side solution for online gaming.

When considering general purpose service delivery across the Internet (not in the context of online gaming), standardization and middleware support are commonplace. Middleware is provided by a number of vendors that aim to ease Internet application development by providing end-to-end solutions to developers. Web Services provide industrywide standardization for describing loosely coupled business-to-business solutions with many server-side solutions supported by popular component-based middleware pertaining to Microsoft's .NET (Fay, 2003) or Sun's J2EE (Sun, 2008a). In addition, the well-established distributed object technology CORBA (Object Management Group [OMG], 2008) (another widely recognized industry standard) has found favor with the industries that require distributed applications that can afford a tighter degree of integration (over that afforded by Web Services).

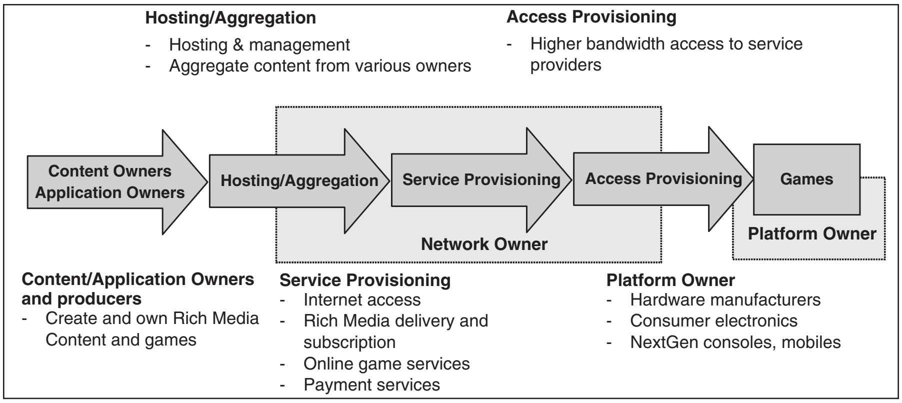

Figure 2. Business value chain for online gaming promoted by IBM

Consideration of Existing Standards

Online games that are server based may appear to lend themselves to implementation using server-side middleware technologies (e.g., .NET, J2EE) or, at the least, using distributed object technologies within a server cluster (e.g., CORBA). Such technologies, with a minor configuration or appropriate usage of additional (readily available) services, are apt at providing general solutions to load balancing, scalability, persistence, and reliability. However, these existing technologies are rarely employed in online gaming solutions. Such services are instead constructed using techniques developed in-house (or at least created by a third party specifically for a particular online game type).

Early Steps

The approaches to middleware standardization for online games so far in the games industry may be considered early steps. No standardization or widely accepted platform promoting interoperability exists today. The XNA approach by Microsoft eases game development and does promote interoperability, but only across Microsoft's own platforms. Sun and IBM have made progress toward a scalable server-side platform solution for online game development but these works may be considered to be in progress[. Most disappointingly, the vast effort put in middleware research and development is yet to be usefully employed by the games industry.

Safe and Useful Content Evolution

Maintaining applications during runtime is a significant research problem in computing science. As persistent virtual worlds are maintained via large clusters of servers

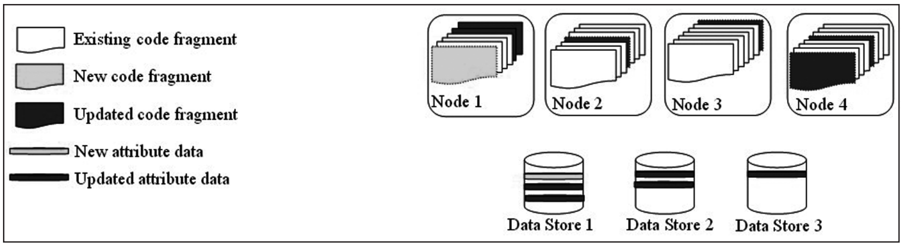

Figure 3. Code fragments and associated attributes

(hundreds, possibly thousands), a need exists to maintain uninterrupted service to players in the presence of server failure, downtime, maintenance, or additions. Achieving this is not straightforward when wishing to simply maintain current service levels. However, the added difficulty of achieving this with a view to increasing virtual-world complexity for enhancing online gaming scenarios places an additional burden on the programmer. This is because such systems are constructed from multiple-code fragments that cumulatively describe the virtual world within which players participate.

Managing Code Fragments

In SECOND LIFE, virtual-world content is described using a scripting language that can be written by players or developers alike. The current object-oriented and component-based approaches to software development encourage such scripting-like approaches allowing functionality and attributes associated to a single virtual-world artifact to be combined into single-code fragments. Such code fragments are then distributed across a server cluster for load balancing purposes with persistent attributes retrieved from the data-store tier as and when required.

Figure 3 presents an example demonstrating the problem of content update and management in the context of code fragments, their persistent attributes, and their deployment across a cluster. In this example, a new code fragment representing an artifact is introduced on Node 1, requiring insertion of new code into an already executing environment and new attribute data to be stored. This results in the need to change other artifact instances (code fragments and attributes) to make best use of the new artifact.

The Role of Reflection

A new code fragment representing an artifact may be manually created. However, the adaptation of the system to accommodate the new artifact should be sufficiently automated to lessen the development burden and ensure safety. To achieve this, the notion of runtime code modification is required. Such an approach to application evolution is commonly termed *reflection*. One use of reflection is to allow the self-reorganization of a system. In essence, reflection may be used to enable self-reorganization of code fragments and associated attributes to allow far-reaching evolutionary change in a safe manner.

There are three increasingly challenging steps required to achieve the evolution described in Figure 3: (a) insert new code fragment, (b) update associated code fragments (locally), and (c) disseminate change across nodes (update remote code fragments).

The work of Rivières, J., Smith B. C., 1984 in the early 1980s is often cited as providing the earliest description of reflection: "A reflective system is one in which otherwise implicit aspects of the system's structure and behaviour are available for explicit inspection and manipulation." To ready a system for "explicit inspection and manipulation," Smith suggested making use of a metalevel. A metalevel assumes responsibility for interpreting computations described in a program. The interpreted program is commonly termed the *baselevel*. As metalevels are also described using programs, they may also be subject to their own metalevel interpretation (in practice, one metalevel and associated baselevel are considered sufficient).

A Mature Discipline

There has been substantial work in the development of reflective systems, allowing programmers using a variety of development environments to make use of a metalevel within their applications: languages with reflection as a cornerstone of their construction (e.g., CLOS—Bobrow, Gabriel, & White, 1991; Smalltalk—Rivard, 1996); enhancing non or partially reflective languages with reflective properties (e.g., C++—Chiba, 1995; Java—Welch & Stroud, 2002); scripting with reflection (e.g., Tcl—Wetherall & Lindblad, 1995); advancements in middleware (e.g., Genie—Bencomo & Blair,2006). Using reflection, programmers have been able to augment their applications with additional qualities (e.g., fault-tolerance—Fabre & Perennou, 1998).

Considering that artifacts are encapsulated within individually separated code fragments and associated persistent attributes, regulating their change via reflective techniques to allow evolutionary change should be possible. Work at Lancaster University (Okanda & Blair, 2003) identified the role that reflection may play in MMOs for satisfying scalability, persistence, and responsiveness requirements. However, the goal of achieving changing of gaming scenarios through content updates is significantly different to the goals of Lancaster's work.

Rules

A major problem with using reflection is that although reflection provides the means for change, it does so with very limited safety constraints. In fact, such an approach is rarely undertaken in industrial applications that exhibit the degree of complexity associated to MMOs. Another alternative, which may be considered a safer option, is to abstract the governance of a virtual world away from the code fragments altogether.

This allows changes to the rules governing gaming scenarios independently from the changes associated to content representation.

In recent years, engineers of eCommerce solutions have begun to make use of rule-based approaches in the construction of their applications (e.g., Oracle, 2007). A number of server-side middleware products now include rule-based tools as part of their application development support (e.g., JBoss, 2008). As business practices are well attuned to operating within particular parameters governed by rules, efforts to construct software tools and techniques to ease the development of business-oriented applications by allowing rules to be clearly stated have preoccupied a number of researchers. By separating the business logic from other aspects of application implementation, one may alter business rules without a requirement to manually update a number of code fragments within the application tier of the server side. In effect, the rules become a clearly identifiable (and manipulative) aspect of an overall application. This has proved successful in the development process, as rules that were not determined accurately at design time could be tailored (or even created) after a system has gone live.

Initially, rule-based software tools originate from work carried out in the artificial intelligence research community. Work carried out in Expert Systems may be considered rule based, with such research eventually taking a number of directions, most prominently, helping create the Business Rule Management Systems in current middleware products. There are a number of rule-based systems available for programmers to make use of, most interesting to MMO developers are those found in distributed systems middleware solutions (e.g., J2EE, .NET). Preliminary studies by Zhu and Morgan (2008) have shown that utilization of rule-based approaches for content evolution may hold promise.

Reflection and Rules in Games Development

No literature or software suggests that the games industry uses rule engines or reflective techniques to manage evolutionary change in persistent virtual worlds. With such a lack of consideration for these two techniques, one may only assume that propriety ad hoc approaches are the mainstay of the game developer. Although reflection may be considered a dangerous programming approach by many, rule-based engines appear quite a convenient abstraction in which to describe gaming scenarios if they can be tailored sufficiently well for use in MMOs. For example, a problem with business rules is that they are quite specific. And what may be needed is a “business rule of thumb,” which can usually be expressed as constraints of various kinds. In this format, they are more accessible for managers to understand and manipulate (more overall governance than isolated rules).

Enabling Rich Interaction

Interest management has been achieved in commercial solutions via the subdivision of the virtual world; players close to each other, sharing the same area, are more likely to interact than those separated by larger distances. Beyond such assumptions, little has been achieved in actually describing influence or interest in a machine-readable format. This makes describing gaming scenarios difficult in the context of the interest and influence expressed by players because of a lack of a language capable for succinctly providing such a description. However, in both academia and industry, there have been attempts in nongaming domains to describe the interaction of participants in a manner suitable for automated analysis.

Describing Services: The Middleware Approach

A first step to achieving a complete language for interest management requires a mechanism for discovery as one must be able to locate gaming scenarios based on their descriptions (in addition to individual objects searching for each other in a virtual world). One area of work that is strongly related to such a problem is that of location and discovery services in distributed systems and middleware research: the ability to query a selection of possible services to determine their suitability for satisfying a given request. This is prevalent in the CORBA Trading Service (OMG, 2000b), which (in the specifications own words) "facilitates the offering and the discovery of instances of services of particular types." In addition to the CORBA approach, the *white*, *green* and *yellow* pages associated to the *Universal Description, Discovery, and Integration* (UDDI) protocol is a mechanism for aiding in the location and discovery of Web Services (UDDI, 2000). As a note to the reader, the UDDI is now quite a different entity (Version 3 in draft form) as the catalog service of the original specification never became popular.

CORBA is a standard that describes the specification of middleware that eases the development of distributed applications. The technology is primarily based on the object-oriented paradigm for constructing applications (as this was, and probably still is, the most popular paradigm for application development). Objects assume the natural unit of distribution in CORBA and may be distributed across nodes in a computer network. The complications associated to accessing an object's methods across a computer network are conveniently handled by CORBA middleware (greatly easing the development of distributed applications). The Trading Service allows an object to register a description pertaining to its functionality. This description can then be searched by potential clients, allowing clients to determine the suitability of different types of objects to satisfy their requests. For example, an object may represent a particular printer with functionality describing the type of printer, pages per second, size of paper, and other additional information that could be used to differentiate between printers. Queries to the Trader Service may assume similar style and presentation as other query languages (such as SQL).

UDDI is the Web Service equivalent of the CORBA Trading Service with a number of differences that reflect the multiorganizational aspects of Web Services. Web Services are designed for loosely coupled application development and, as such, provide only service interfaces (without the concept of instantiation). The standard for Web Services specifies the basic protocol and textural representation of messages with additional services to aid distributed application development.

Agreeing on Service Usage

The actual game play scenario description of a successful interest management service requires a language capable of describing interaction. Research in the area or eCommerce has produced a number of solutions for describing the interaction of services for the purposes of monitoring and evaluation. The need to automate the regulation of service usage across the Internet to ensure that service consumers and service providers adhere to predefined rules of interaction has resulted in a number of technologies. The fact that interaction must be described in an unambiguous manner indicates that such technology may be suitable for exploitation within interest management. The bases for such technologies are *Service-Level Agreements* (SLAs).

SLAs specify the *quality of service* associated with the interaction between the provider of a service and a service consumer. SLAs are gaining in importance as increasing numbers of companies conduct business over the Internet (e.g., banking, auctions), requiring the positioning of SLAs at organizational boundaries to provide a basis on which to emulate the electronic equivalents of contract-based business management practices. A number of languages have been developed to describe SLAs, with such languages integrated into overall systems to provide complete monitoring/evaluation solutions for eCommerce.

Conclusions

This article has described four research challenges that are inhibiting the advancement of enabling technologies suitable for creating highly interactive online gaming scenarios:

- *Engineering practices:* The competitive nature of the games industry has resulted in propriety approaches that are not conducive to promoting standardized middleware for online games.
- *Longevity:* Persistent virtual worlds are maintained in a way that limits the evolution possible within gaming scenarios, restricting players to repetitive game play scenarios.
- *Describing game play:* A lack of a descriptive language to successfully describe game play inhibits the ability to automate analysis and creation of gaming scenarios and the support of such scenarios.

Commercial Success Hides Technological Shortcomings

The commercial success of online gaming in recent years has masked the technological inadequacies highlighted in this article. Although rapidly growing in popularity, online games are not the most significant revenue stream of the games industry at the time of writing this article. In the future, one may envisage that online gaming may become the dominant force in the games industry. However, to achieve this position of dominance, the games industry needs to consider more diverse gaming scenarios

Table 1. Engineering Issues in Online Gaming

Engineering Issue	Games Most Affected	Hindrance	Possible Solutions
Propriety play	All games	Inability to share content across gaming environments	Adoption of middleware standards
Longevity	MMOs	Not possible to adapt existing gaming scenarios beyond trivial fixes and expansions	Reflective programming combined with rule engines
Describing play	MMOs	Not possible to describe and reason about gaming scenarios and model all behavior	Utilization of discovery services combined with language describing service provision

Note: MMO = Massively Multiplayer Online.

to satisfy the gaming requirements of a wider variety of players than exists at the moment; commercially successful online games tend to be dominated by role-playing games, mostly related to the fantasy genre, that provide game play that differs little.

The challenges highlighted in this article go some way to identifying what direction current online games research could take. Table 1 presents a brief summary of the discussion presented here.

Consider Technology Beyond Gaming

The challenges described here have been related to work achieved, both in the game industry and other computing disciplines. When considering a broader field of computing science, a clear picture emerges that work from a variety of disciplines may be appropriate to the advancement of online game development. Therefore, a requirement for the online game developer is a need to extend their view of network-related technologies and prevent reinvention of existing techniques in the guise of online gaming middleware.

Promotion of Standards

The competitive nature of the gaming industry needs to be tempered to encourage collaboration and agreement on core technologies. In the development of distributed applications, there has long been an agreement as to how technologies may interact via adherence to industry standards. An understanding that collaboration via standardization, although not ideal for every requirement, provides a basis that reduces the development and maintenance costs of software and promotes interoperability. If the cost of persistent virtual-world development, deployment, and maintenance remains high, then the opportunity for innovation will be hampered.

Glossary

3D Studio Max Popular modeling tools for 3D content creation (also used to describe the file format associated to this modeling tool)

CORBA Common Object Request Broker Architecture, a technology to ease distributed application development in heterogeneous environments (different programming languages and operating systems) using the object-oriented paradigm

DirectX Suite of libraries to allow game development on Microsoft platforms (also used to describe the file format associated to DirectX 3D models)

First-Person Shooter Nonpersistent gaming environment characterized by fast, real-time, action, and low number (less than 50) of players participating at any one time

Grid An approach to computing, where a collection of loosely coupled computers collaborate to perform a task

High-Level Architecture (HLA) Architecture designed to accommodate the creation of distributed simulations

Interest Management Mechanism for describing interest and influence in distributed virtual worlds

J2EE A programming platform to ease server-side development in the Java language

Middleware Software residing above the operating system and below the application layer that eases distributed application development

MMO Commonly used abbreviation of MMORPG (Massively Multiplayer Online Role Playing Game) usually with persistent qualities (e.g., WORLD OF WARCRAFT, WARHAMMER ONLINE)

NET A development framework from Microsoft providing an array of libraries and code to ease windows application development

OMG Object Management Group, standards body responsible for CORBA and related standards

UDDI Universal Description, Discovery, and Integration attempt to catalog services available across the Internet (strongly related to Web Services)

Region Area of virtual world that identifies one or more areas of localized game play interaction

Shard A duplication of a virtual world's static environment (term originally used in database terminology, where a shard is individual separately stored rows)

SLA Service-Level Agreement, embodiment, in an electronic form, of an agreement defining service usage and provision between machines

Web Services Standards to promote interoperability, primarily for machine-to-machine communications (usually associated with interorganizational computing)

XNA Set of runtime tools for easing the development of computer games from Microsoft

Acknowledgments

I would like to thank my reviewer's and a number of colleagues for providing recommendations that have improved this article immeasurably. In particular, I would like to thank Rob Smith, Rob Minson, Pierre Corbeil, and David Crookall.

Declaration of Conflicting Interests

The author declared no conflicts of interest with respect to the authorship and/or publication of this article.

Funding

The author received no financial support for the research and/or authorship of this article.

References

Ang, C. S. (2006). Rules, gameplay, and narratives in video games. *Simulation & Gaming, 37,* 306-325.

Bencomo, N., & Blair, G. (2006, October). *Genie: A domain-specific modelling tool for the generation of adaptive and reflective middleware families*. Paper presented at the Sixth OOPSLA Workshop on Domain-Specific Modelling, Portland, OR.

Birman, K. P. (1986). *Isis: A system for fault-tolerant distributed computing* (Tech. Rep., UMI Order Number TR86-744). Ithaca, NY: Cornell University.

Bobrow, D. G., Gabriel, R. P., & White, J. L. (1991). CLOS in context: The shape of the design space. *Communications of the ACM, 34*(9), 28-38.

Carlsson, C., & Hagsand, O. (1993). DIVE: A platform for multiuser virtual environments. *Computers and Graphics, 17,* 663-669.

Carpenter, A. (2003, June 11). Applying risk analysis to play-balance RPGs. *Gamasutra*. Retrieved June 24, 2009, from http://www.gamasutra.com/features/20030611/carpenter_01.shtml

Chiba, S. (1995, October). A metaobject protocol for c++. In *Proceedings of the 10th ACM Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA)* (pp. 285-299), Austin, TX.

CNet. (2006). *World of warcraft battles server problems*. Retrieved September 2, 2008, from http://news.com.com/World+of+Warcraft+battles+server+problems/2100-1043_3-6063990.html

EA. (2008). *Medal of honour: Airborne assault*. Retrieved September 2, 2008, from http://www.ea.com/moh/airborne/

Fabre, J. C., & Perennou, T. (1998). A metaobject architecture for fault-tolerant distributed systems: The FRIENDS approach. *IEEE Transactions on Computers, 47,* 78-95.

Fay, D. (2003, April). *An architecture for distributed applications on the Internet: Overview of Microsoft's .NET platform*. Paper presented at the International Parallel and Distributed Processing Symposium (IPDPS'03) PP90a, Nice, France.

Fong, G. (2006). Adapting COTS games for military experimentation. *Simulation & Gaming, 37,* 452-465.

JBoss. (2008). *JBoss Drools project*. Retrieved September 2, 2008, from http://www.jboss.org/drools

Kushner, D. (2005). Engineering everquest. *IEEE Spectrum Magazine, 42*(7), 34-39.

Linden. (2007). *Linden Lab, security and second life.* Retrieved September 2, 2008, from http://blog.secondlife.com/2006/10/09/security-and-second-life

Linden. (2008). *Second life*. Retrieved September 2, 2008, from http://secondlife.com/

LoTR. (2008). *Lord of the rings online*. Retrieved September 2, 2008, from http://moria.lotro.com/

Lu, F., Parkin, S. E., & Morgan, G. (2006, October). *Load balancing for massively multiplayer online games*. Paper presented at NETGAMES 2006, ACM SIGCOMM, Singapore.

Lucas Arts. (2008). *Trials of Obi-Wan*. Retrieved September 10, 2008, from http://starwarsgalaxies.station.sony.com/trialsofobiwan

Miller, D. C. (1996, March). *The DOD high level architecture and the next generation of DIS*. Paper presented at the Workshop on Standards for the Interoperability of Distributed Simulations, Orlando, FL.

Minson, R., & Theodoropoulos, G. (2008, May). Push-pull interest management for virtual worlds. In *Proceedings of the 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing (ISORC),* IEEE Computer Society (pp. 189-194), Orlando, FL.

Morgan, G., Lu, F., & Storey, K. (2005, April). Interest management middleware for networked games. In *Proceedings of the I3D 2005 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (pp. 57-63), Washington, DC.

Noy, A., Raban, D., & Ravid, G. (2006). Testing social theories in computer-mediated communication through gaming and simulation. *Simulation & Gaming, 37,* 174-194.

Okanda, P., & Blair, G. S. (2003). The role of structural reflection in distributed virtual reality. In *Proceedings of the VRST 2003* (pp. 140-149), Osaka, Japan.

Object Management Group. (2000b). *Trading object service specification*, OMG (Object Management Group) technical specification. Retrieved September 2, 2008, from http://www.omg.org/docs/formal/00-06-27.pdf

Object Management Group. (2008). Retrieved September 2, 2008, from Object Management Group's CORBA Website: http://www.corba.org/

Oracle. (2007). *Oracle business rules: Technical overview* (An Oracle white paper). Retrieved September 2, 2008, from http://www.oracle.com/technology/products/ias/business_rules/index.html

Postal, J. (Ed.). (1981). *Internet protocol: DARPA Internet program protocol specification* (RFC 791). Arlington, VA: USC/Information Sciences Institute.

Powel, E. T., Mellon, L., Watson, J. F., & Tarbox, G. H. (1996). Joint precision strike demonstration (JPSD) simulations architecture. In *Proceedings of the 14th Workshop on Standards for the Interoperability of Distributed Simulations* (pp. 807-810), Orlando, FL.

Prakash, E., Brindle, G., Jones, K., Zhou, S., Chaudhari, N., & Wong, K. (in press). Advances in game technology: Software, models and intelligence. *Simulation & Gaming*.

Roccetti, M., Ferretti, S., & Palazzi, C. E. (2008, May). The brave new world of multiplayer online games: Synchronization issues with smart solutions. In *Proceedings of the 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing (ISORC),* IEEE Computer Society (pp. 587-592), Orlando, FL.

Rivard, F. (1996, April). Smalltalk: A reflective language. In G. Kiczales (Ed.), *Proceedings of Reflection'96* (pp. 21-38), San Francisco.

Rivières, J., Smith B. C., 1984. The implementation of procedurally reflective languages, In Proceedings of the 1984 ACM Symposium on LISP and Functional Programming, (pp. 331-347), Austin, TX.

Shaikh, A., Sahu, S., Rosu, M.-C., Shea, M., & Saha, D. (2006). On demand platform for online games. *IBM Systems Journal, 45,* 7-19.

Sharp, C. (2004). *IBM middleware to enable new business models in the online games industry*. Retrieved May 5, 2004, from http://www-106.ibm.com/developerworks/webservices/library/ws-intgame

Sun. (2008a). *Java 2 platform, enterprise edition (J2EE) overview*. Retrieved September 2, 2008, from http://java.sun.com/j2ee/overview.html

Sun. (2008b). *Sun Microsystems "Project Darkstar."* Retrieved September 2, 2008, from http://www.projectdarkstar.com/

UDDI. (2000). *UDDI technical white paper*. Retrieved September 2, 2008, from http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf

Waldo, J. (2008). Scaling in games & virtual worlds. *ACM Queue—Game Development, 6*(7), 10-16.

Welch, I., & Stroud, R. (2002). From Dalang to Kava the evolution of a reflective Java extension. In *Proceedings of the 16th European Conference on Object-Oriented Programming* (ECOOP 2002), No. 2374, Lecture Notes in Computer Science (pp. 205-230), Malaga, Spain.

Wetherall, D., & Lindblad, C. J. (1995, July). *Extending Tcl for dynamic object-oriented programming*. Paper presented at the Tcl/Tk Workshop '95, Toronto, Ontario, Canada.

White, W., Koch, C., Gehrke, J., & Demers, A. (2008). Better scripts, better games. *ACM Queue—Game Development, 6*(7), 18-25.

WoW. (2008). *World of warcraft*. Retrieved September 2, 2008, from http://www.worldofwarcraft.com

XNA. (2008). *XNA Creators Club*. Retrieved September 2, 2008, from http://creators.xna.com/

Zenke, M. (2008, June/July). Land of fire: The rise of the tiny MMO. *Game Developer Magazine, 15*(6), 7-11.

Zhu, L., & Morgan, G. (2008, November). *Runtime evolution for online gaming: A case study using JBoss and Drools*. Paper presented at the Game Design and Technology Workshop, Liverpool, UK.

Bio

Graham Morgan is a faculty member at Newcastle University, where he leads research and teaching in the area of video games. He has published many articles about video gaming and enjoys participating in the academic community: conference keynotes, chairing conferences, program committees, and journal editorials. Many of his past students are now employed in the video games industry. Most recently, he spent 2008 to 2009 at George Mason University (the United States) helping them create their first video game degree. Contact: graham.morgan@ncl.ac.uk.