Database Research in Computer Games

Alan Demers
Cornell University
ademers@cs.cornell.edu

Johannes Gehrke
Cornell University
johannes@cs.cornell.edu

Christoph Koch
Cornell University
koch@cs.cornell.edu

Ben Sowell
Cornell University
sowell@cs.cornell.edu

Walker White
Cornell University
wmwhite@cs.cornell.edu

ABSTRACT

This tutorial presents an overview of the data management issues faced by computer games today. While many games do not use databases directly, they still have to process large amounts of data, and could benefit from the application of database technology. Other games, such as massively multiplayer online games (MMOs), must communicate with commercial databases and have their own unique challenges. In this tutorial we will present the state-of-the-art of data management in games that we learned from our interaction with various game studios. We will show how the issues involved motivate current research, and illustrate several possibilities for future work.

Our tutorial will start with a description of data-driven design, which is the source of many of the data management issues that games face. We will show some of the tools that game developers use to create and manage content. We will discuss how this type of design can affect performance, and the data structures and techniques that developers use to ensure that the game is responsive. We will discuss the problem of consistency in games, and how games ensure that players all share the same view of the world. Finally, we will examine some of the engineering issues that game developers have to deal with when interacting with traditional databases.

This tutorial is intended to be self-contained, and provides the background necessary for understanding how databases and database technology are relevant to computer games. This tutorial is accessible to students and researchers who, while perhaps not hardcore gamers themselves, are interested in ways in which they can use their expertise to solve problems in computer games.

Categories and Subject Descriptors

D.3.2 [**Programming Languages**]: Language Classifications—*Specialized application languages*

General Terms

Languages, Performance

Keywords

Games, Scripting, Aggregates, Indexing

SIGMOD'09, June 29–July 2, 2009, Providence, Rhode Island, USA.
ACM 978-1-60558-551-2/09/06.

1. DESCRIPTION

Data-Driven Game Design

Many modern computer games are authored using *data-driven* development techniques [3]. In data-driven development, the game content is separated as much as possible from the game software, and placed in auxiliary data files. For a lot of game content — such as music and art — this is unsurprising. The game industry has standardized file formats and tools for managing this kind of data. However, game content can also include many things that we think of as software, such as character behavior and triggers for in-game events. As a result, game companies spend a lot of effort building custom content creation tools and scripting languages to support this development model.

Game studios embrace data-driven design for multiple reasons. First of all, it allows them to easily partition the work between the software engineers and the designers. Software engineers program the abstract game engine, while game designers create content and character AI for a specific game. Designers often have very little programming experience, so they need tools that are tailored specifically to their natural workflows. In addition to accommodating designers, data-driven approaches allow game companies to amortize their development costs over multiple titles. Game expansion packs typically contain new content, but they include very few modifications to the underlying software.

Data-driven design is also good for players, as it can significantly increase interest in a game title and improve user experience. For example, *World of Warcraft* contains an XML specification language that allows players to define the look of their user interface, from window positions to button functionality [14]. This allows players to customize their game to improve their "productivity". Some games like *Second Life* go further and provide users with a complete scripting language that they can use to create new content [7]. This type of user-generated content can greatly extend the playable lifespan of a popular game.

Data-driven design creates large amounts of data that game engines must manage efficiently. To understand the challenges that this data presents, we first have to understand how this data is created and modified. Hence this tutorial begins with an overview of the state-of-the-art in content creation tools for games. Our presentation will focus primarily on scripting languages for both single player and massively multiplayer games. We will show the ways in which designers manage XML files, define character behavior, specify event triggers, and create game-specific server functionality.

Performance Challenges

While programming game behavior outside of the application software has many development advantages, it often comes at a performance cost. If designers are not careful, they can easily write scripts where every object in the game interacts with every other object, resulting in computations that are $\Omega(n^2)$ in the number of game objects. This a real concern in game development, and some studios have taken drastic measures — such as removing support for iteration and recursion from their scripting languages — to keep their designers from producing computationally expensive behavior [10]. As scripts are sometimes processed every animation frame, seemingly innocuous code can cripple the performance of a game.

As with databases, game developers often rely on indices to speed up computations that involve relationships between pairs of objects. Many games use traditional spatial indices such as BSP trees or Octrees. However, games also have many unusual spatial data structures that may not be familiar to a database audience. For example, navigational meshes are used to represent the ways in which a character is allowed to move about the geography [12]. Furthermore, like many data structures in games, these meshes are often annotated by a designer or technical artist to include extra semantic information — such as whether a position is a good hiding place or is easily defensible.

We will also look at how game developers have been using parallel programming to improve performance; this is an area in which game developers potentially have a lot to learn from the database community. Indeed, many of the techniques that game programmers have been using to optimize physics calculations and other types of computations on GPUs look very similar to the techniques that database engines use for join processing [1]. Furthermore, there is a growing body of research into game engine APIs and scripting languages to simplify this type of development [11, 9, 13].

Consistency Challenges

In the case of MMOs, *consistency* is often just as important as performance. Just as with a database, games require that their data — which is often the state of the entire world — be in a consistent state. Ensuring this consistency is difficult, however, because players are performing conflicting actions at a very high rate. This means that traditional approaches such as locking transactions are often too slow for games. Furthermore, scripting languages almost never have support for concurrency, and game designers and players may not have the background to take advantage of these features effectively. Indeed, concurrency violations in scripting languages are one of the largest sources of bugs and exploits in MMOs [6]. In this tutorial, we will present several ways in which games are dealing with this challenge, and how we can use this to inform future research.

One of the more successful solutions has been the use of "causality bubbles". This term refers to a variety of techniques where games predict which players may issue conflicting interactions with one another and dynamically partition their databases to reduce server load. Several commercial games use a variant of this method developed for very specific environments [2, 4]. For example, EVE online runs a continuous differential equation that takes into account the acceleration of every space ship in a solar system. This differential equation allows them to determine, for any given time interval, which ships can move within range of each other; this way they can dynamically partition the map into feasible units. More recent research has attempted to generalize this idea to arbitrary transactions [5].

Another way in which games deal with concurrency is by having weaker consistency guarantees. Sometimes this means ensuring that world is consistent at only a very coarse level; animation or other uncontested activity in the game may be out of sync between computers but the persistent game state is the same. Other games go even further and allow players to have inconsistent, but very similar game states. For example, "aggro management" is the technique that *World of Warcraft* uses to target opponents and process combat. It assigns abstract roles to the participants, which allows the game to handle combat without exact spatial fidelity.

Engineering Challenges

Many online games already use databases, and they run into the traditional problems of database application development. For example, they need to ensure that the bridge between the client software and the SQL code is robust enough to handle changes in each. However, there are several interesting challenges that are unique to games, and which motivate future research. In this tutorial, we will examine some of these issues as time permits.

One engineering issue is the problem of *intelligent checkpointing*. MMOs use commercial databases for persistence and to recover from server crashes. However, players interact with the game so fast that it is too expensive to process every single action with the database. Most games have an in-memory database layer that processes all actions, and only writes to the database periodically. In some games, these checkpoints can be as far as 10 minutes apart [8]. Recoveries may force a player to repeat a difficult fight or lose a particularly desirable reward. As a result, games need ways to checkpoint intelligently, writing to the database when important events are completed, and not just at regular intervals.

Legacy schemas are another major engineering issue in games. Some MMOs, like the venerable *Everquest*, have game worlds that have been active for almost a decade. To remain vibrant and competitive, these worlds continually add new features and abilities. These new features often require schema changes in the world database. Schema migrations on a live system can be very painful for game developers. They often choose to write data as an unstructured "blobs" into a single attribute, so that they can preserve their old schemas [8]. Until game developers have better migration tools, they constantly have to balance database support with sustainability.

2. PRESENTER BIOGRAPHIES

The tutorial is based on material from all of the authors, but it will be presented by Johannes Gehrke, Ben Sowell, and Walker White.

Johannes Gehrke is an Associate Professor in the Department of Computer Science at Cornell University. Johannes' research interests are in the areas of data mining, data privacy, complex event processing, and computer games and

virtual worlds. Johannes has received a National Science Foundation Career Award, an Arthur P. Sloan Fellowship, an IBM Faculty Award, the Cornell College of Engineering James and Mary Tien Excellence in Teaching Award, and the Cornell University Provost's Award for Distinguished Scholarship. With Raghu Ramakrishnan, he co-authored the undergraduate textbook Database Management Systems (McGrawHill (2002), currently in its third edition), used at universities all over the world.

Ben Sowell is a graduate student in the Department of Computer Science at Cornell University. His research is in scaling data-driven games and simulations. His graduate studies are supported by an NDSEG fellowship.

Walker White is the Director of the Game Design Initiative at Cornell, an interdisciplinary undergraduate program training students in the design and development of computer games. Walker has ties to the games industry, having consulted for several game studios and led a roundtable at the Austin Games Developer Conference. He is also a researcher in the Cornell BigRedData Group and has authored several papers on data stream processing and data management in computer games. His current research interests are in data management and data-driven design for computer games.

3. ACKNOWLEDGEMENTS

This research is supported by the National Science Foundation under Grant IIS-0725260, the Air Force under Grant FA9550-07-1-0437, a grant from Microsoft Corporation, and by New York State Science Technology and Academic Research under Agreement Number C050061. Any opinions, findings, conclusions or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the sponsors.

4. REFERENCES

[1] E. Coumans. Physics tutorial: Parallel game physics for spu. In *Proc. GDC*, San Francisco, CA, 2008.

[2] B. Dalton. Online gaming architecture: Dealing with the real-time data crunch in MMOs. In *Proc. Austin GDC*, Austin, TX, September 2007.

[3] M. DeLoura, editor. *Game Programming Gems*, volume 1. Charles River Media, 2000.

[4] Halldor Fannar Guðjónsson. The server technology of EVE Online: How to cope with 300,000 players on one server. In *Proc. Austin GDC*, 2008.

[5] N. Gupta, A. Demers, J. Gehrke, P. Unterbrunner, and W. White. Scalability for virtual worlds. In *Proc. ICDE*, 2009.

[6] T. Keating. Dupes, speed hacks and black holes: How players cheat in MMOs. In *Proc. Austin GDC*, Austin, TX, September 2007.

[7] Linden Labs. Linden Scripting Language. http://wiki.secondlife.com/wiki/LSL_Portal.

[8] J. Lee, R. Cedeno, and D. Mellencamp. The latest learning - database solutions. In *Proc. Austin GDC*, Austin, TX, September 2007.

[9] Sun Microsystems. Project darkstar. http://www.projectdarkstar.com/.

[10] S. Posniewski. Massively modernized online: MMO technologies for next-gen and beyond. In *Proc. Austin GDC*, Austin, TX, September 2007.

[11] B. Sowell, A. Demers, J. Gehrke, N. Gupta, H. Li, and W. White. From declarative languages to declarative processing in computer games. In *CIDR*, 2009.

[12] P. Tozour. Building a near-optimal navigation mesh. In *AI Game Programming Wisdom*, volume 1, pages 298–304. Charles River Media, 2002.

[13] W. White, A. Demers, C. Koch, J. Gehrke, and R. Rajagopalan. Scaling games to epic proportions. In *Proc. SIGMOD*, pages 31–42, 2007.

[14] WoWWiki. XML user interface. http://www.wowwiki.com/XML_user_interface.