INSTRUMENTATION OF VIDEO GAME SOFTWARE TO SUPPORT AUTOMATED CONTENT ANALYSES

T. Bullen and M. Katchabaw
Department of Computer Science
The University of Western Ontario
London, Ontario, Canada
N6A 5B7
tbullen@uwo.ca, katchab@csd.uwo.ca

N. Dyer-Witheford
Faculty of Information and Media Studies
The University of Western Ontario
London, Ontario, Canada
N6A 5B7
ncdyerwi@uwo.ca

KEYWORDS

Content analysis, automated content analysis, software instrumentation, Unreal Engine.

ABSTRACT

Content analysis of video games is an important process that supports many business, policy, social, and scholarly activities related to the games industry. Unfortunately, collecting the large quantity of data and statistics required for content analyses tends to be an incredibly arduous task. Supports are clearly necessary to facilitate content analysis procedures for video games.

This paper introduces an approach to automating content analyses for video games through the use of software instrumentation. By properly instrumenting video game software, content analysis procedures can be either partially or fully automated, depending on the game in question. This paper discusses our overall approach to instrumentation and automation, as well as our experiences to date in instrumenting Epic's Unreal Engine, providing sample results from early experiments conducted to date. Results have been quite positive, demonstrating great promise for continued work in this area.

INTRODUCTION

Content analyses of video games involve coding, enumerating, and statistically analyzing various elements and characteristics of games, including violence, offensive language, sexual activity, gender and racial inclusiveness, and so on. While content analysis has limitations, as demonstrated in (Holsti 1969; Newman 2004), it is invaluable in providing a quantitative assessment of games to complement more qualitative analyses, as recently suggested in (Bogost, 2006). As such, content analysis is an important tool to scholars of game studies and other media issues; policy makers dealing with issues of regulation, ratings and censorship; psychologists dealing with media effects; developers and publishers producing games; and parents, educators and game players using these games.

Unfortunately, problems arise when one applies traditional content analysis procedures, for example from television or film, to video games. These procedures are manual and tend to be time consuming and labour-intensive, resulting in problems such as either limited play-time, sometimes just the first level (Heintz-Knowles et al. 2001) or first few minutes (Brand and Knight 2003), or, alternatively, playing very few games to have time for more thorough examinations (Grimes 2003). Traditional analyses often do not consider the effects of player interactivity and non-linearity in games, which can limit their accuracy unless these issues are explored more fully. These issues are further compounded by the rapid rate at which games are released and the medium evolves; it becomes quite difficult to conduct thorough analyses of a reasonable portion of games with the limited time and resources typically available for doing so. A solution to these problems is clearly needed.

This paper introduces the concept of automating content analysis of video games. This approach addresses the above problems by taking advantage of the fact that, unlike other forms of media, video games are ultimately software executing on a computing device. Content analysis can be partially automated by having other software on the computing device monitor game execution and collect and report the data traditionally collected using manual procedures. Full automation may also be possible in some cases by having software take the role of the player and generate gameplay experiences without human intervention. In providing these supports, automation effectively reduces the time, labour, and resources required to conduct a thorough content analysis. This allows longer and more representative analyses of more games, and allows analyses to be conducted more frequently. Automation also permits broader studies of interactive and non-linear play, with the potential for more data to be collected than through manual processes alone.

To automate content analysis, our current work uses a framework of instrumentation to augment games in a minimally invasive fashion to collect the necessary data and exert control over the game to conduct a thorough analysis. As proof of concept, we have used our framework to instrument Epic's Unreal Engine (Epic Games 2005), a popular engine used in the development of numerous games. Through instrumenting the engine, we are able to automate the content analysis of any game developed for the engine. In particular, this paper presents experiences from content analysis experiments conducted on Unreal Tournament 2004 (Digital Extremes 2004).

The remainder of this paper is organized as follows. We begin with a discussion of our approach to instrumentation and automation for content analyses. We then describe our implementation and proof of concept work with Epic's Unreal Engine. We then discuss our experiences in conducting simple content analysis experiments on Unreal Tournament 2004. Finally, we conclude this paper with a summary and a discussion of directions for future work.

INSTRUMENTATION FOR CONTENT ANALYSES

Software instrumentation is a concept new to video games, but has been used for several years in other types of software to enable the collection of data and the exertion of control over the software. The basic premise is to embed additional code into the execution stream of an application to enable these data collection and control activities. The approach taken in this work is derived from our earlier work in the area (Katchabaw et al. 1999) with updates as necessary to support the needs of video game software.

Instrumentation Architecture

The instrumentation architecture used in our current work is depicted in Figure 1, and discussed in detail in the remainder of this section.

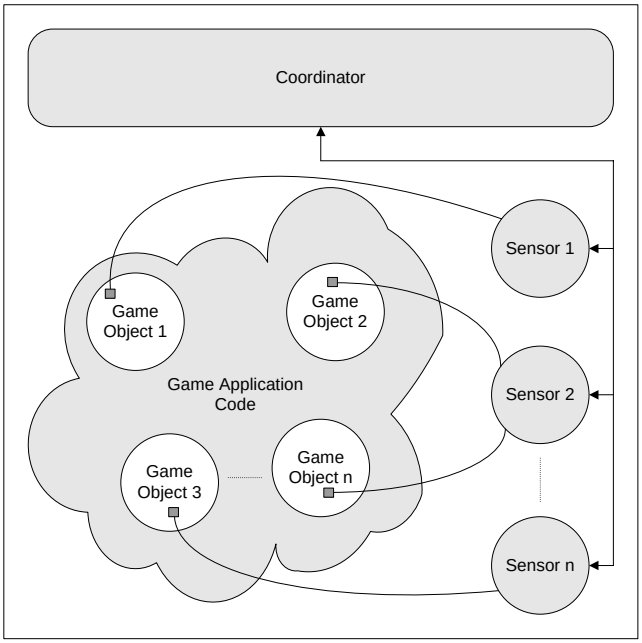

Figure 1. Instrumentation for Content Analysis

Game Application Code

Game application code refers to the original source code from the game that is being instrumented. It is composed of a collection of objects that work together to deliver the functionality of the game. By gathering data and statistics from the appropriate game objects at the right times, we can conduct an effective quantitative content analysis of the game as it is being played.

Sensors

Sensors are instrumentation components that are used to collect, maintain, and (perhaps) process information to be used in content analyses. Sensors interface with objects in the game application code through probes that are inserted into the game. Such probes are typically macros, function calls, or method invocations that are placed in the execution stream of an object's source code during development, or are event listeners listening for events emitted by the object as its code executes. Sensors typically reside in the same address space as the game application code, perhaps executing in separate threads. Depending on the game and how it is constructed, however, sensors could theoretically exist in separate processes.

Sensors can be used to collect a wide variety of measurements useful to a content analysis. This includes instances of violence (type of violence, source and target of violence, result of violence), offensive language (what was said, source and target of the language), character demographics (race, age, gender), and so on. Sensors can also collect a variety of game and game world information, including the game being played, the type of game, the level of the game, the time played, and so on.

For flexibility, sensors can also have their behaviours tuned, in some cases at run time. This includes whether they are active or not, what data is being collected, how data is processed, how data is being reported, and so on.

Coordinator

The coordinator is an instrumentation component that is responsible for directing the content analysis activities occurring within a game. This includes initializing and configuring sensors, processing reports of collected data and statistics from sensors, and handling clean-up activities when the game terminates. The coordinator is also the point of contact for tuning behaviour of sensors and other aspects of content analysis at run-time. Like sensors, the coordinator also typically resides in the same address space as the game application code, but could be located in a separate process, depending on the game in question.

Instrumentation Operation

When a game instrumented for content analysis is launched, one of its initialization activities before play commences is to create a coordinator to initialize the instrumentation. This, in turn, creates the required sensors, and configures them to collect data as required for the content analysis in question.

As the game executes, probes for the sensors will gather the information needed as they are either invoked in the execution stream of the corresponding game objects, or in response to events generated by the game objects, depending on the structuring of the game application code in question. This information is accumulated and processed by the sensors and either reported to the coordinator as it is collected or stored for further processing and reporting in the future. Any such reports received by the coordinator are logged to a file, or presented or recorded as deemed necessary by whoever is conducting the content analysis.

When the game is completed, or is otherwise terminated, the coordinator flushes out any pending reports and

deactivates and destroys all sensors. At this point the coordinator itself shuts down, and the game terminates.

PROTOTYPE IMPLEMENTATION

As a proof of concept, we have used our instrumentation framework to instrument Epic's Unreal Engine (Epic Games 2005) to enable content analyses. We chose to instrument an engine because engine-level instrumentation enables us to conduct content analyses of all games built on top of that engine without requiring instrumentation on a game-by-game basis. The Unreal Engine is also a popular engine among developers and hobbyists, providing a good collection of games for study in the future.

Since we were targeting the Unreal Engine in this work, our instrumentation was developed using UnrealScript. While a C or C++ instrumentation library is preferable to provide support across a variety of games and game engines, most game engines used in industry do not provide code-level access to their engines or only do so in a cost-prohibitive fashion, including the Unreal Engine. UnrealScript fortunately provided all the access that was required for our content analysis instrumentation.

Adding our instrumentation for content analysis to the Unreal Engine was fairly straightforward, as shown in Figure 2. Each Unreal game type has a Game Info object that defines the game in question. Among other things, this object contains a collection of game rules defining various aspects of how the game is played, and a collection of mutators. Mutators, in essence, allow modifications to a game and gameplay while keeping the core elements and game rules intact.

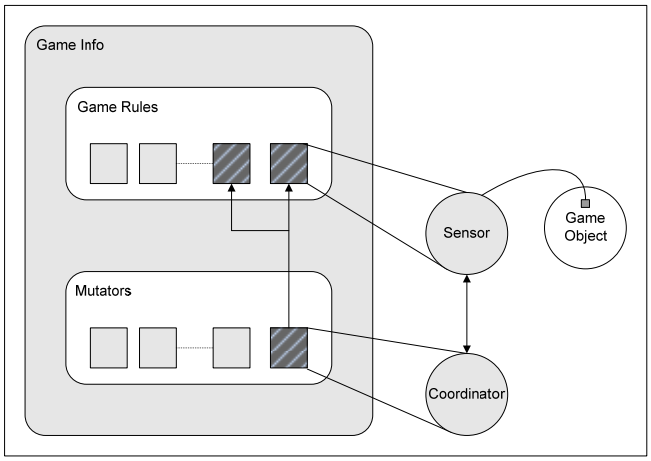

Figure 2. Instrumenting Epic's Unreal Engine

Our instrumentation is loaded into a game in the form of a special content analysis mutator. This mutator contains the instrumentation coordinator, as described in the previous section. When loaded, the coordinator in this mutator spawns an appropriate collection of sensors to gather the information required for content analysis. Each sensor is contained within a game rule that is appended to the list of game rules contained within the Game Info object by the instrumentation coordinator. In doing so, the sensors are able to access the stream of events generated by the various game objects in the game, and extract the required information to conduct the content analysis.

For example, suppose we were to conduct a content analysis on a game and were interested in tracking the deaths that occurred within the game. When the content analysis mutator is loaded, the coordinator contained within the mutator creates a new game rule containing a sensor capable of measuring and tracking deaths in the game. This rule is then appended to the list of rules for the game. As the game executes, the sensor in the game rule waits for events indicating that a death has occurred within the game. When a death occurs, the sensor observes the event and updates its internal statistics, perhaps by pulling additional information in from other objects in the process.

Data collected by sensors can either be reported as it is collected, or in the form of summaries reported when the game is completed or terminated. The method used depends on the needs of the particular content analysis taking place. Unfortunately, the Unreal Engine does not provide a fully functional file access mechanism at the UnrealScript level. However, the Unreal Engine does provide several logging capabilities which are quite sufficient for generating reports of game activities for content analysis.

The Unreal Engine allows mutators to be selected, configured, and loaded by the user at run-time, which is a very useful feature. This allows content analysis to be enabled and disabled dynamically at run-time, and allows the user to tailor and fine tune various elements of the content analysis easily. For example, the user can choose which types of data to collect and not collect, and can tailor various elements of the collection and reporting processes.

To date, sensors have been implemented to collect a variety of information required for a thorough content analysis. This includes death of game characters, use of offensive language, gender and racial diversity in characters, and a variety of game details such as time played and so on. Sensors to collect other information are currently under development.

EXPERIENCES AND DISCUSSION

In this section, we describe our initial experiences in using our Unreal-based prototype system for simple content analysis experiments, and discuss observations made in conducting these analyses.

Experiences with Unreal Tournament 2004

To validate our prototype implementation, we needed an Unreal-based game that would use our instrumented Unreal Engine as its foundation. For our purposes, we used Unreal Tournament 2004 (Digital Extremes 2004), as it is one of the most popular Unreal-based games, and it was readily available at our disposal. Unreal Tournament 2004 is a first-person shooter game that supports a wide variety of

different game types and sets of game rules, individual and team-based games, and single player, multiplayer, and spectator modes of play. (In spectator mode, games can be played with no human players, and the game's display is used to observe the game's progress.) Consequently, there are many gameplay options provided within this game.

The test system used for experimentation was a dual-core 3.0GHz Pentium D system, with 2GB RAM, a 250GB hard drive, and an ATI X1800 graphics accelerator card. The operating system in this case was Microsoft Windows XP SP2. As such, the test system exceeded the recommended system requirements for Unreal Tournament 2004.

With this experimental environment, we conducted several content analysis experiments using a variety of game configurations. This included the following:

- Standard deathmatch (single player and spectator)
- Team deathmatch (single player and spectator)
- Onslaught (single player and spectator)
- Capture the flag (single player and spectator)

The standard deathmatch game is an individual game, while the other modes were all team based games, with artificial intelligence-controlled non-player characters filling the rosters of teams. Levels played were chosen randomly, and team size and other characteristics as appropriate were set at the levels' default values.

Summary results from one experiment are provided in Figure 3, showing that the content analysis instrumentation works as expected, collecting all of the required data. As a result, the instrumentation appeared to be quite effective in facilitating quantitative content analysis procedures. Furthermore, this instrumentation was able to provide all required data and statistics with minimal additional work required by the user. (All that was necessary was to activate the content analysis mutator on its first use, and to collect reports from the generated log file upon completion of the game. After activating the content analysis mutator, it remains active for every game until it is deactivated.)

Further Discussion

Our initial testing and experimentation with our content analysis instrumentation yielded several interesting observations worthy of further discussion and examination.

Quality of Data

While conducting experimentation with our content analysis instrumentation, we felt it important to verify the accuracy of collected data with more traditional manual procedures using a human observer watching gameplay sessions. In doing so, it was found that the statistics computed by the instrumentation matched those computed using the manual procedures.

Interestingly enough, the statistics computed by the instrumentation appeared to be more complete and more accurate as the pace of the game and positioning of in-game cameras at times made manual procedures error-prone and frustrating. Instrumentation was also able to capture both on-screen and off-screen activities, and distinguish between the two, which is difficult, if not impossible, to accomplish using manual procedures alone.

```
------------Level Info------------
Level Name: Rrajigar
Game Type: DeathMatch
Total Players: 14
AI Players: 13
Human Players: 1
Spectators: 0
Male Players: 13
Female Players: 1
Level Loaded: 0:26:45
Game Finished: 0:30:29
Gameplay Elapsed (Seconds): 240.88
AI Dialog: 28
Human Dialog: 27
-----------------------------------

----------All Player Stats--------
Total Deaths: 47
Total Suicides: 1
Total Kills: 46
Total AI Deaths: 46
Total Human Deaths: 1
Total Deaths Caused By AIs: 22
Total Deaths Caused By Humans: 25
Total Female Deaths: 12
Total Male Deaths: 35
Total Deaths Caused By Females: 6
Total Deaths Caused By Males: 41
-----------------------------------

--------Local Player Stats--------
Player Deaths: 1
Player Suicides: 0
Player Killed: 1
Deaths Caused By Player: 25
Player Killed By AI: 1
Player Killed By Human: 0
Player Killed By Male: 1
Player Killed By Female: 0
AI Deaths Caused By Player: 25
Human Deaths Caused By Player: 0
Female Deaths Caused By Player: 7
Male Deaths Caused By Player: 18
Deaths Witnessed By Player: 29
-----------------------------------

------------Team Info-------------
Not A Team Game
-----------------------------------

------------Expletives-----------
ass: 2
-----------------------------------
```

Figure 3. Sample Summary of Content Analysis Data from an Unreal Tournament 2004 Game

Quantity of Data

Another observation deals with the quantity of data collected and how this data is reported. Increasing the amount of data available to a content analysis has the potential to increase its accuracy and the amount of insight that can be obtained from the analysis. Our content analysis instrumentation was found to be able to generate reports with considerable detail, and the elimination of manual collection procedures allows data to be collected from more gameplay sessions than previously possible.

Unfortunately, increasing the quantity of data handled by instrumentation has the potential to increase processing and storage requirements, as this data must be collected, stored, and reported for use in content analysis. As a result, there is a risk of negative impacts on the performance of the game if the quantity of data collected is too high, or if it is reported so frequently that it interrupts the flow of the game. While we could measure no change in performance during our experimentation, this could be an issue in some content analyses. For example, in our experiments, we tracked violence in terms of character deaths. Instead of this, suppose violence was tracked in terms of the number of shots fired by weapons in the game or the number of shots hitting a character. This would result in a much higher quantity of data being collected, stored, and reported at a faster rate, and this could have an impact on the performance of the game.

Consequently, one must be careful in tuning the quantity of data collected for a content analysis. This issue requires further study.

Partial versus Fully Automated Content Analyses

Another interesting observation came when comparing partially automated content analyses to fully automated analyses. A partially automated analysis requires a human player to drive the game while the embedded instrumentation handles the data collection and reporting activities, whereas a fully automated analysis requires no human player, with the game essentially driving itself using artificial intelligence-controlled non-player characters.

Since Unreal Tournament 2004 supports a spectator mode in its game sessions, it is possible to conduct a fully automated content analysis on the game, simply by having artificial intelligence-controlled non-player characters play the game by themselves. Unfortunately, these games can take significantly longer than games involving human players, as the non-player characters tend to be less effective at achieving victory than human players. Also, since the skill level of non-player characters are more balanced, the kills in a game can be more evenly distributed in the absence of a dominant human player, requiring more kills in total to end a game. For example, consider the standard deathmatch game whose summary is shown in Figure 3. The human player clearly dominated this game, scoring more than half the total kills in the entire game, and quickly bringing the game to an end in reaching the kill limit set as a victory condition. When played in spectator mode with the same number of non-player characters in the same level, the game took nearly three times as long to complete on average, and the average number of kills per non-player character was over sixteen times higher. With the human player no longer dominating, the game results were substantially different.

This indicates that the nature of data collected during a partially automated content analysis might differ significantly from a fully automated analysis. Since a partially automated analysis involving a human player is likely a more accurate reflection of an actual gameplay experience than a fully automated analysis, this raises questions about the suitability and validity of fully automated analyses. However, since a fully automated analysis removes the need for human interaction with the game, this kind of analysis is still attractive as it is less resource intensive, allows data to be collected from more game sessions, and removes bias and unwanted effects introduced by the human players of the game. Consequently, this issue also requires further study.

CONCLUSIONS AND FUTURE WORK

Content analysis plays several important roles to the video games industry, but is unfortunately an arduous task to complete in an accurate and thorough fashion. The content analysis instrumentation introduced in this paper has the potential to greatly facilitate content analyses of video games through partially or fully automating the process. A prototype implementation of this instrumentation in Epic's Unreal Engine has been demonstrated through experimentation with Unreal Tournament 2004 to effectively assist in content analyses, and shows great promise for the future.

There are many possible directions for future work in this area. Based on the success of initial content analysis experimentation, more thorough and detailed analyses should now be conducted on Unreal Tournament 2004, combining quantitative data collected through our instrumented engine with more qualitative observations. Experiments should also be expanded to include more Unreal-based games, as well other game engines, to provide further validation of our instrumentation. As mentioned in the previous section, further study is required into tuning content analysis instrumentation to maximize the quality, accuracy, and thoroughness of results, while at the same time minimizing the impact on game performance. Additional testing and experimentation is also required to study the advantages and disadvantages of partially automated analyses compared to fully automated analyses.

REFERENCES

Bogost I. 2006. *Unit Operations: An Approach to Videogame Criticism*. Cambridge, Mass.: MIT Press.

Brand J. and K. Knight. 2003. "The Diverse Worlds of Computer Games: A Content Analysis of Spaces, Populations, Styles and Narratives." *In the Proceedings of DiGRA 2003: Level Up.* Utrecht: University of Utrecht. (November).

Digital Extremes. 2004. *Unreal Tournament 2004 – Editor's Choice*. (August).

Epic Games. 2005. *Unreal Engine 2, v. 3369.* (December).

Grimes S. 2003. "You Shoot Like a Girl: The Female Protagonist in Action-Adventure Games." *In the Proceedings of DiGRA 2003: Level Up.* Utrecht: University of Utrecht. (November).

Heintz-Knowles K., J. Henderson, C. Glaubke, P. Miller, M. Parker, and E. Espejo. 2001. *Fair Play: Violence, Gender and Race in Video Games*. Oakland, California: Children Now. (December).

Holsti O. 1969. *Content Analysis for the Social Sciences and Humanities*. Reading: Addison-Wesley.

Katchabaw M., S. Howard, H. Lutfiyya, A. Marshall, M. Bauer. 1999. Making Distributed Applications Manageable Through Instrumentation. Journal of Systems and Software, 45 (1999).

Newman J. 2004. *Videogames*. New York: Routledge.