Extending Game Participation with Embodied Reporting Agents

Dan Fielding, Mike Fraser, Brian Logan and Steve Benford
School of Computer Science and IT
University of Nottingham
Nottingham NG8 1BB, UK
{dgf, mcf, bsl, sdb}@cs.nott.ac.uk

ABSTRACT
We introduce a multi-agent framework to generate reports of players' activities within multi-player computer games so that other players who are currently unable to participate can keep track of the activities of their colleagues. We describe an initial implementation of our framework as an extension to the Capture the Flag game within Unreal Tournament. We report the results of a preliminary experiment that shows that embodied reporter agents give varying coverage depending on deployment strategies used, and, in particular, suggests that the dynamic assignment of reporter agents by an editor agent can provide more effective coverage than static assignment schemes. Finally, we explore future applications of this work including other genres of games, the emergence of games as spectator sports, implications for pervasive games as well as non-gaming applications.

Categories and Subject Descriptors
I.2.11 [**Artificial intelligence**]: Distributed Artificial Intelligence – *Intelligent agents, Multiagent systems*; Applications and Expert Systems – *Games*.

General Terms
Design, Experimentation, Human Factors.

Keywords
Reporting, game agents, on-line participation, audiences.

1. INTRODUCTION
Recent years have seen an explosion in the field of on-line gaming. From chess, to multi-player shooters, to massively multiplayer online fantasy environments, players are now logging on from across the globe to pit their wits against one another on a simulated battlefield. Agent technologies are at the centre of this explosion, especially when it comes to creating engaging and believable non-player characters. In this paper, we explore another role for agents in online computer games - automatically generating reports of the action so that external observers can keep track of the action from a distance.

There are two primary motivations for this idea. First, we recognise that online gaming is a highly social activity. Gaming is not just about winning, but is also about comradeship and community and we therefore wish to develop services that support players in maintaining contact and coordinating their activities with fellow players, even though they may be distributed around the globe. Second, emerging massively-multiplayer games provide persistent experiences that continue around the clock - even when a player is not present - and in which gamers invest great effort in building up a character over a long time period. These players may wish to receive news from the game even when they are unable to play, or may wish to be alerted to important new developments that require them to return to the game at short notice (an increasingly likely scenario as such games become accessible using mobile technologies making it easier to quickly step into and out of a game). A further motivation for this work is the recognition that games are beginning to emerge as a spectator sport, as evidenced by the growth of game tournaments, professional players and early examples of television shows that broadcast multiplayer gameplay, which raises requirements for new ways of portraying games to external viewers who are not directly participating.

In this paper we describe a multi-agent system to address the challenges involved in reporting, editing and presenting game information to external participants, in order to support more complex forms of participation in online gaming. These challenges include being able to observe and reason about the activities of human players, deciding which activities are relevant to an external participant as well as how to present them, and also providing appropriate mechanisms for players to maintain privacy or conversely, to deliberately try to become visible to others.

2. KEY PRINCIPLES
We begin by motivating our approach to multi-agent reporting. Development of the framework has been driven by two core principles:

1. Agents that capture information about a game should be directly embodied within the game so that they are visible to and subject to the same constraints as players.
2. Distributing the responsibility for the different functional roles of extracting, filtering and reporting information between different kinds of agent can provide the kind of

ACE'04, June 3-5, 2004, Singapore.

flexibility that is needed to deal with the unpredictable nature of events within computer games.

Considering our first principle, one approach to reporting on an online computer game would involve developing a single omniscient agent to both gather and deliver information about an environment to external observer. In order to have complete knowledge of all of the events taking place within the game, an omniscient agent would need to be implemented as a privileged system process that would have access to all of the information passing through a game server. While this may be a good approach in some circumstances, most notably where global coverage is needed, the game world is small and the agent can be trusted by the players to be impartial and not to give information away to competing players, we have adopted an alternative approach based on embodied agents. In this case, agents which collect information are part of the game world and, like the players themselves, can only perceive a limited part of the total environment. The agent is also directly visible to the players and, unless specifically protected, is subject to the normal interactions of the game (e.g., might be attacked by the players). We prefer this approach for several reasons.

Our primary focus is on providing information about a game to individuals who may in time become players. These individuals may have different interests (e.g., in following particular players or teams) which might best be serviced by dedicated agents. At the same time, it is important not to give these potential players an unfair advantage by giving too much information away (in which case, omniscient agents could become a potential security threat by revealing strategies to the opposition).

Reporting on people's online activities raises the important issue of privacy, especially for larger games that extend beyond small groups of friends. Players may not wish everything that they do - and especially say - to be reported to a wider audience. Embodied agents have both limited ability to gather information due to their constrained perspective and perhaps more importantly, are visible to the players who will therefore be more aware of when they are being watched and can adapt their actions accordingly. For example players can try to avoid the agents, can modify their behavior when around them, or can literally fight to maintain their privacy by destroying them. Conversely, players can deliberately try to influence the agents by approaching them, either 'acting up' for the camera (e.g., celebrating a victory) or perhaps targeting specific messages at particular individuals the outside world. In either case, embodiment gives the players some influence over what is reported.

Finally, from a technical perspective, the larger and more complex the environment that an agent is charged with reporting on, the harder it becomes to cover all of the events taking place. Although omniscience avoids the need for the agent to position itself appropriately so as to see the events as they occur, the requirements of data collection, (especially for widely distributed games that span multiple game servers) and the inferences required to produce interesting commentary (e.g., to select the most relevant events) may ultimately limit the scalability of the omniscient approach. A more scalable approach is likely to involve multiple cooperating agents where no single one agent has complete knowledge of the game. A single monolithic commentator may also be less easy to develop and maintain than a modular system comprising several types of individual agents working together.

This brings us to our second core principle. The overall task of reporting on a computer game involves a variety of tasks. One of these is gathering information, where embodied agents have to make decisions about what information to gather and where and how to find it. A second is presenting this information to external observers, which may be done in a variety of ways - ranging from animated talking reads through to simple text messages - depending upon the devices they are using and their current situation. There is also a need to decide how to coordinate these various functions, for example determining how many news gathering agents can be sent into the game world before their presence has an adverse affect on the game and given that their number will be limited, deciding how best to deploy them. Given the complexity of these tasks, we follow an approach in which the responsibilities for different aspects of reporting are distributed across different agents that then collaborate in a scalable and flexible way.

3. KEY PROBLEMS

In this section, we briefly outline some of the problems which must be addressed to perform the key tasks of obtaining information, processing information, and presenting information.

3.1 Obtaining Information in the Game World

In order to report on world events, an embodied agent needs to maneuver into a suitable position to observe or otherwise detect them. First, the agent needs some way of being in the right place at the right time – either by working out where that may be for itself, or by being told where to go by another agent. This in turn implies that (some) agent must be capable of predicting where events are likely to occur, or that the agent can react quickly enough to get to an event observed by another agent before it finishes. Second, the agent needs to be able to actually navigate its way to the intended destination, preferably utilising the route involving lowest “cost” – this could be interpreted as the route that is quickest, safest, or maybe some other criteria.

Once in a position, the agent needs to produce some form of useful output about the events in the game, and continue to do so until the event is finished. We assume that, in general, the game server does not generate “predigested” events suitable for reporting. The agent therefore needs to deduce which events are occurring based on the raw sensory data that it receives, and work out what is happening in situations where this is not immediately obvious, perhaps taking an educated guess if there is some degree of uncertainty. The agent also needs to identify when an event starts and stops, in order to know when to both start and stop reporting, and also to know when it is a suitable time to move on to a new event. If the event changes location while in progress, which is often the case when an event is centred around moving object such as a particular player, the agent needs to follow the object of interest and maintain line-of-sight. While witnessing the action, an agent needs to obtain all relevant information and produce its report without interfering with the gameplay too much, e.g., getting in the way or getting shot, if possible. This could involve watching from a distance, and perhaps taking evasive action if imminent danger is sensed.

Finally, an agent needs some way or prioritising the events that it is possible to view. It needs to decide which events that it witnesses in the game world are interesting enough to warrant being passed on in a report, either to viewers or other agents. It also needs to be able to be able to consider whether to abandon viewing an event in order to search for one of greater interest,

considering the time it will take to relocate and the possibility of missing an event occurring at the location that it's already watching.

3.2 Processing Information

By assumption, each reporting agent has access to only limited information about events in the game. They are also likely to have only limited inferential capabilities. As a result, they are fallible, at least in the sense that they can misinterpret what the observe or otherwise distort its significance. With a system comprising multiple agents, we could utilise some form of report collation to validate and verify reports. The agents could also share information on where the most interesting events in the world are currently occurring, in order to redirect agents that are currently watching nothing of interest. Incomplete or conflicting reports could be used to direct the collection of additional related information, in order to clarify uncertain situations.

Furthermore, there could be several layers of information processing. For example, in a situation where there is a tournament of individual games taking place, each game could have its own set of agents gathering information from the game and delivering the collated event reports to an audience interested in that one particular game. At a higher level, there could be another agent taking these reports and collating them, producing a report on the state of the tournament as a whole without including the kind of in-depth information about each game that the agents at lower levels produce.

3.3 Presenting Information

With all the event information collected and processed, the agents are left with the task of presenting it to the outside world somehow. The exact process depends on a number of criteria, such as:

- When to present the information. We could have a commentary produced during the game, or a post-game report and analysis, for example, each requiring a different approach.
- The output device. Reports delivered using a device such as a web-browser or IRC-channel can be much more verbose than a report delivered using, for example, SMS on mobile phones.
- How to organise and present the information. There is the possibility of using plain text, or adding in pictures and possibly video feeds, or even using a talking-head style virtual presenter.
- The interests of the viewers. Some viewers could be interested in a particular player or team, for example, or a particular type of event within the game.

In many cases, it may be useful to be able to produce two or more forms of output at the same time and/or to be able to tailor reports to the user's current situation. For example, we could imagine a scenario in which each user has their own presenter agent which adapts as the user moves from high to low bandwidth devices or to situations in which the commentary should be less intrusive. This implies that it would be useful for a commentary system to support different forms of presentation being swapped in and out easily.

3.4 Related Work

A number of systems have attempted to address these problems. For example, there is a large amount of related work on commentary systems in general, including systems using single [1] or multiple [2] agents.

There is also considerable body of work on using external agents to assume roles in virtual environments, such as characters in interactive fiction [3] or artificial players in games [4]. Existing tools designed to deliver reports from game platforms to external media such as an IRC channel or a webpage (e.g. TTM [5]), typically run on the game server itself, in effect performing the same role as an omniscient, disembodied reporter to gather and deliver event information. There is also a body of work on delivering commentary on Robocup [6] games, e.g. the MIKE commentary system, which uses multiple distributed agents to analyse and commentate [7]. (The MIKE system has also been adapted for other situations.) With regard to the earlier discussion of issues in detection of interesting events, there is some related work on the use of various forms of group detection to infer interaction between avatars, such as the use of clusters to extract interesting scenes in a recorded event [8]. In previous work, we have investigated the use of F-formations [9] to detect social encounters [10].

However, so far as we are aware, no single system has addressed all of the reporting challenges outlined above. Nor has any of the work to date addressed the issue of embodiment and the problems of privacy seriously. In the next section, we outline a new framework for game reporting which attempts to address these issues.

4. REPORTING FRAMEWORK

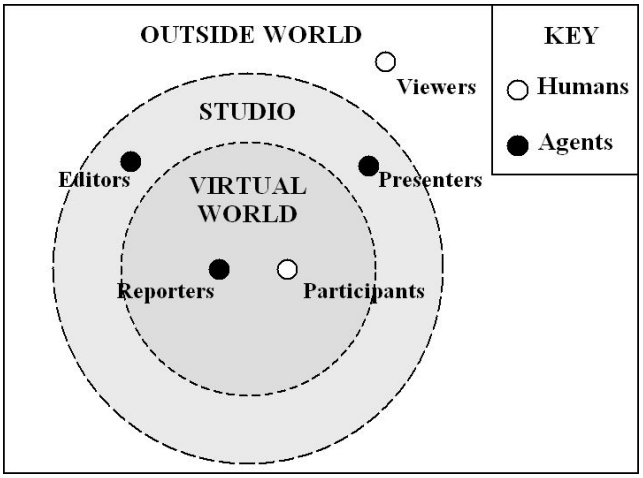

Figure 1. Reporting Framework.

Figure 1 summarises our framework in terms of a number of defined roles that have been inspired both by conventional human news-gathering techniques and the key problems discussed in the previous subsection. Participants are those inhabitants of the virtual world who can directly influence the course of events, and thus are the individuals that are interesting to report on. In our chosen example, which we describe in more detail in the next section, our virtual world is a map in a game of Unreal Tournament, and the Participants are the players' avatars inside the game.

Reporters also inhabit the virtual world, but cannot directly influence it in the same manner as the Participants. Reporters gather information about the state of the world, attempting to deduce what events are taking place and passing the information they gather on to Editors. Following our first core principle, Reporters are embodied and have a constrained perspective.

Editors take the information they receive from Reporters, and pass on the most important and relevant data to Presenters. Editors can attempt to guide the Reporters in their information gathering, possibly by trying to co-ordinate their activities to improve the coverage of the game or by notifying Reporters of the types of events they are most interested in receiving reports on.

Presenters are responsible for delivering the information received from the Editors to the outside world, in whatever format is desired. For example, we might realize a presenter that updates a webpage as events take place, or that delivers news reports as an animated talking head, or one which delivers short notifications of important events to users via mobile phone SMS. The time at which commentary is delivered is also an important consideration; some presenters could be tasked with delivering real-time commentary to an audience as events unfold, while another presenter could be responsible for producing a summary once the events in the virtual world have completed. More sophisticated reporters could re-order information to make it more easy to understand, more dramatic, or even omit events which they feel uninteresting.

Editor and Presenter agents could also potentially be embodied in the virtual world, although this is at least initially of less concern than the Reporters being embodied. Editors and Presenters are not necessarily designed to directly interact with or observe the Participants in the game environment. We might however embody such agents in cases where the behind-the-scenes reporting environment itself was interesting for study or scrutiny, such as in observations of production techniques and reporting itself, or possibly in future games that could include news feeds from elsewhere in the game world or where players might even try to capture and control the news service as part of the game.

In the remainder of this paper, we concentrate primarily on Reporter agents as these provide the most basic service of capturing information about the game upon which Editors and Presenters subsequently build. However, in section 4, we show that the introduction of an editor which dynamically allocates roles to reporters can improve the coverage of events reported, and provides some support for our multi-agent framework. In the next section, we describe the details of the game environment used to test our approach.

5. SYSTEM IMPLEMENTATION

The prototype consists of a variable number of embodied reporter agents and a single (non-embodied) editor agent. The reporters observe events in the game world, make simple inferences about their significance and send reports to the editor. The editor collates the reports and sends its output to a log file to allow us to analyse the performance of the system. The editor may also direct the reporters' activities in a general way, e.g., by sending them to a particular region of the game world.

Our prototype system is based on “Capture the Flag”, one of the game types provided by Unreal Tournament [11]. Unreal Tournament (UT) is a 'first person shooter' game, in which players compete in a map or level to achieve specific game objectives. While the games in UT have only a small number of players and are often quite short, they retain key characteristics of games like Everquest [12], while still remaining manageable for development and testing. The UT game engine has rich and well-documented APIs capable of supporting agent interaction and the game code itself is robust and well maintained. Significant parts of the game logic are exposed in UnrealScript making it easier to tailor the game environment for our experiments. In common with many other recent first person shooter games, UT offers "bot" players: simple computer-controlled avatars who are designed to fill in for absent human players. UT's built-in bots can play Capture the Flag games fairly competently, and simplify initial system testing, allowing games to be played and reported upon without the need to find human players to participate in the game.

Capture the Flag is a team-based game and was chosen as it was felt that team-based games offer a richer set of situations and events for reporting than individual-based games, as well as engendering the possibility that spectators might affiliate with one another in favour of a particular team.

In Capture the Flag, two teams of players (Red and Blue), each attempt to collect the opposing team's flag from their base and carry it back to their own team's flag base to score a point, while at the same time preventing the enemy team from doing the same to their flag (typically by shooting them). There are five flag-related events that we wish to detect, as they are relevant to the outcome of the game:

- Flag Takes: When a player collects the enemy team's flag from their flag-base.
- Flag Drops: When a player carrying the enemy team's flag is killed. The flag lies on the ground at the location of the player's death.
- Flag Pickups: When a player collects the enemy team's flag from a dropped location.
- Flag Returns: When a player collects their own team's flag from a dropped location. When this happens, the flag is instantly returned to the flag-base.
- Flag Captures: When a player carries the enemy team's flag to their own flag in their own flag-base. This event causes the player's team to score a point.

The game is won by the team that performs the most flag captures (and thus scores the most points) within a set time limit. Individual players are also awarded points for favorable actions, e.g., killing opposing players and retrieving their own flag from enemy hands. While these would be interesting to report, individual player scores are unrelated to team scores, and as such are not reported in our current prototype.

5.1 Modifications to Unreal Tournament

We extended the Capture the Flag game in a number of ways to support reporting.

We modified the game to allow a third, impartial team (Green) to join the game in addition to the two standard player teams (Red and Blue). Players on the Green team are embodied but are not allowed to directly influence the game - they cannot collect items or use weapons. Normally, players are rewarded with points for killing an enemy player. The built-in UT bots will not fire upon reporters intentionally but can inadvertently kill or injure a

reporter when shooting at a player on the opposing team. Since reporters are neutral spectators and not directly involved the game, we need some mechanism to discourage the (human) players from firing upon them at will. We therefore introduced a penalty in the form of a points deduction for any player that kills a reporter. If the game server is set so that friendly players cannot injure one another then neither team can injure the reporters. For the purposes of our tests, however, we have chosen to leave 'friendly fire' enabled, allowing us to test the abilities of editors to reassign reporters to cover for those killed while witnessing events.

We also modified the way in which players communicate within the game. Unreal Tournament features two modes of text-based communication: Global and Team. Whenever a player messages the Global channel, all players in the map can read what they have to say, and when a player messages the Team channel, only the player's teammates can read the message. We have added a proximity requirement to the Global form of text chat, so that one must first stand near to the player in order to hear what they are saying. The purpose of this proximity-based text messaging is to force the reporters to be close to any players that they wish to hear talking. Chat sent to the Team channel can still be heard regardless of position by the player's teammates, but if an opponent or reporter is standing near the player who is talking then they can "overhear" them.

We use the Gamebots [13] interface to allow agents to communicate with the UT game server. Gamebots is a socket based interface which allows agents running on remote machines to connect to the game world and interact with it by sending commands for desired actions to the server and receiving sensory information from it. The agents were developed using the SIM_AGENT toolkit [14, 15].

5.2 Reporters

Figure 2. Annotated screenshot showing an embodied reporter and a player within Unreal Tournament.

The reporters connect to the UT server via Gamebots. Gamebots provides each reporter with data that approximates to that available to a player. A reporter's sensory range is limited, and to obtain information about events in other parts of the map, the reporters must physically move to a different location. The reporters navigate around the UT map using the built-in pathnodes system, which enables bots (and agents) to move around the map without performing calculations on the map geometry itself.

5.2.1 Reporting Events

By remembering the objects in the game world that they have sensed in the past and the state of objects that they can currently sense, the reporter can infer which actions are being carried out within the game. For example, if a reporter observes a dropped flag, it assumes that the last player it saw carrying the flag dropped it. When reporters are killed they "respawn" near to a randomly determined flag-base. Respawing causes the reporter to forget the current state of the flags and its current position (since it may end up a large distance away from where they killed, and thus a large distance away from the task they were performing). It also gives the players a way of influencing what the reporter 'knows' and hence can report.

The five flag events can be detected by the reporter(s) in different ways. For example, some events can be detected from more than one location at the same time while others require a reporter to be present at one particular location in the world.

5.2.2 Event Location

Flag captures can be detected globally, as they change the game score and this is something that Gamebots agents are always notified of. However, in order to correctly work out which player performed the flag capture, the reporter must witness them doing so. If they fail to witness the flag capture event, they can take an informed guess at who performed the event based on the last player they saw carrying the flag.

Flag returns can be detected in one of two locations. First, if a reporter witnesses a player performing a flag return at the location of the dropped flag (by walking over it), that is obviously enough to produce a complete report of that event. Second, however, is the possibility of a reporter watching the flag-base from which the flag was taken, and to which it is being returned; if the reporter sees the flag reappear in the flag-base and the game score has not changed (since then it would likely be a flag capture instead), it can deduce that it was returned, although it will be unable to know who performed the flag return event.

Flag drops can be detected by a reporter witnessing a flag lying uncarried outside of a flag-base. In order to know who dropped the flag, the reporter must either have seen it occur, or infer that it was dropped by the player last seen carrying the flag.

Flag pickups and takes can be detected either by a reporter witnessing the event happen (i.e. where the player walks over the flag), or by witnessing a player carrying the flag. In the latter instance, it is often unclear whether the flag was taken (from a flag-base), or picked up (from outside a flag-base), so the reporter will take a guess based on the last known location of the flag.

5.2.3 Event Duration

The events the reporters currently detect are notionally instantaneous. However, in many cases, the fact that an event occurred can be inferred from the state of the game world for a short time after the event actually occurred.

The flag capture event itself is typically detected as soon as the game score changes. In order to detect the player performing the event, the reporter must spot said player carrying the flag before the capture takes place. Depending on where the flag is collected from by the player, this can either be quite a brief period of time (e.g. if the flag was dropped near to the player's flag-base), or a long period of time (e.g. if the flag was carried all the way from

its flag-base), and the chance of a given reporter witnessing the player carrying the flag varies depends on this duration. Flag pickups and takes are similar, detectable as long as the player is carrying the flag.

Flag drops can be detected during the period the flag is lying unattended outside of the flag-base. However, to be able to report which player dropped the flag, the reporter must have observed some player carrying the flag since it was last returned to a flag base.

Flag returns can be detected whenever the reporter notices that a flag which was once missing has since reappeared in its flag-base, so long as the game score didn't change. However, in order to detect the player who performed the flag return, the reporter must witness that player walk over the dropped flag. Since such an occurrence is instantaneous, this leads to flag returns being possibly the hardest event to produce a complete report on.

5.2.4 Roles & Coordination Strategies

Reporters have three basic roles they can fill within our framework: idle (i.e., roaming the map at will); watching a flag base; and pursuing a flag carrier. If a reporter which is watching a flag base sees a player take the flag from the flag base, it switches to the "pursuing a flag carrier" role. The pursuit role itself has a number of sub-roles, e.g., a dropped flag is considered more interesting that a carried flag. In all roles, the reporters observe and report the five flag events listed in section 4. They also report player deaths while pursuing a flag carrier and the death of any player whose name appears in a list of "interesting" players. This allows basic tailoring of the interests of the reporters. Reports are sent via the Team message channel and can be directed to other reporters, editors or all members of the Green (reporting) team.

We investigated three reporter coordination strategies. In the first, each reporter wanders around the environment looking for and following events of interest. In the second strategy, the reporters are assigned static roles for the duration of the game, e.g., watching a flag base. In the third strategy, an editor agent dynamically assigns roles to reporters based on the information it receives from the reporters about events in the game and the current state of their reporters, e.g., their current location, or whether they have just been killed.

5.3 Editor

The prototype system contains a single editor. The editor agent has two main responsibilities.

First, the editor needs to pass any interesting segments of the output generated by the reporters to the presenter. Since reporters are not infallible, it is beneficial to verify this data before passing it on, for example by removing conflicting reports or by requiring multiple reporters to detect the same game event. The editor classifies reports into one of three categories: unconfirmed, confirmed, and conflicting. Unconfirmed reports are those produced by a single reporter, confirmed reports are those produced by multiple reporters who are in agreement with one another, and conflicting reports are those produced by multiple reporters who do not agree on all of the information; for example, two reports may give a different instigator for an event, or may even disagree on the type of event itself in the case of flag takes being mistaken for pickups and vice versa. Depending on how much emphasis the editor places on generating entirely accurate commentary, it may choose to send unconfirmed reports to the presenter, or alternatively it may choose to await confirmation from other reporters, passing on information about that event to the presenter only when it has been reported on more thoroughly.

Second, in the case in which the reporter are following a dynamic role allocation strategy, the editor must ensure that all the reporters are gathering information that the audience deems interesting. The editor attempts to assign roles to reporters in such a way as to provide good coverage of the events in the game, and tries to avoid having reporters standing idle or assigning multiple reporters to the same task unnecessarily. With two or more reporters, the editor attempts to keep one reporter watching each flag base at all times. Whenever a reporter assigned to one of these tasks becomes unable to perform it any longer - due to the reporter being killed, becoming disconnected from the game, or pursuing a flag carrier - the editor selects the most appropriate replacement to fill the vacant role. In our current implementation, the most suitable replacement is defined as the closest idle reporter to the point at which the task is carried out. The current prototype focuses mainly on the assignment of roles to reporters; the collation and verification of reports is still under development.

While the game is in progress, the editor also periodically publishes reports on flag captures in the form of an html document.

5.4 Log Parser

The UT game server logs all flag related events that occur in the course of a game. The reporters produce output in the same format as the game server logs. We have developed a tool to parse these log and report files, comparing events which actually took place in the game to those recorded by the reporters. In order for an event detected by a reporter to match an event in the game's log files during comparison, the reporter must correctly identify not only the type of event but also the player who was performing that event. In some cases it is possible that only the event will be detected and that the reporter will be unable to determine who performed it. In the case of such a partial match, or if a reporter attributes an event to the wrong player, we treat this as a half match; that is, if all events in a single game were detected in such a way, the coverage rate for the reporter in that game would be given as 50%. The log parser allows us to get an accurate indication of how complete the reporters' coverage of events is - the greater the percentage of events detected by the reporters, the better - and thus detect when alterations to the system are beneficial

6. RESULTS

We have conducted a series of experiments to discover how well embodied reporting agents can cover these events in a game. We arranged three sets of games, using automated players and reporter agents. In each set of games the agents used a different strategy for reporter coordination. In the first case, our reporters moved around the environment following events of interest, with no inter-reporter coordination. In the second case, reporter agents were assigned 'static' roles, i.e. were employed on a particular task throughout the game. For example, a single agent might attempt to watch a flag base at all times. The third case used an Editor agent to coordinate and assign roles between reporters dynamically, determining at particular points what task each particular reporter should be undertaking. Our hypothesis was that the first case (no role allocation) would produce worse

coverage of events than the second case (static role allocation), and that the second case (static roles) would produce worse coverage that the third (dynamic role allocation).

The following graphs depict the results from the three series of tests, showing the coverage of various configurations of reporters and editors in our game environment. Each test used a sample of 50 simulated games, and the graphs show the average coverage of one to four reporters across the five flag events for these games. Games were each 20 minutes long, with six bots acting as players (split into two teams of three each) on the "Adept" skill setting.

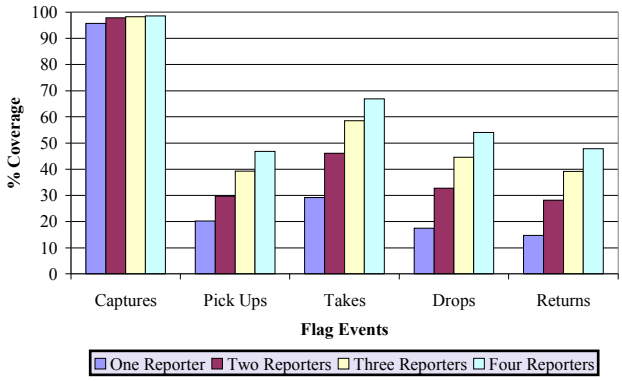

Figure 3. Results: Reporters with No Roles.

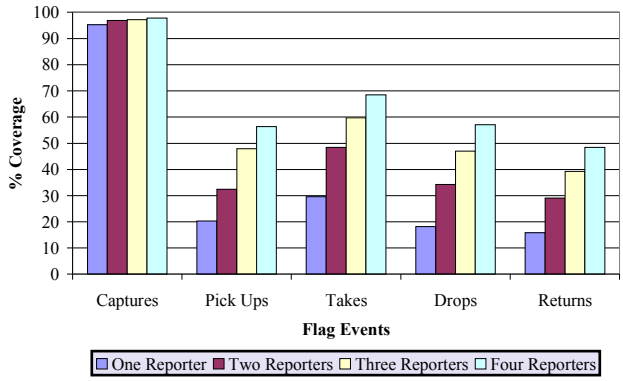

Figure 4. Results: Reporters with Fixed Roles.

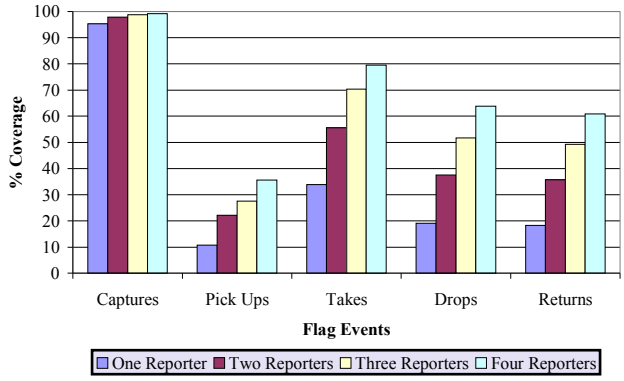

Figure 5. Results: Reporters with Dynamic Roles.

In all three sets of results, the flag capture event coverage is higher than the other event types. There are two reasons for this. Firstly, whenever a team's score count increases, the reporters can surmise that a flag has been captured, because there is no other way for a team to score. Secondly, in order for a player to capture the enemy team's flag, they have to carry the flag from one flag-base to the other. Since the flag-bases are usually at opposite ends of the map, it is likely that the reporters will spot them and be able to correctly work out who it was that captured the flag, especially on maps where there are few routes between the two flag-bases. Because the flag capture event coverage relies so little on the position of the reporters, it remains consistently high for the sets of results shown.

Figure 3 shows the coverage achieved by the reporters when none were given any specific role; all reporters assumed an 'idle' state and roamed the map by walking between flag-bases. Aside from the flag capture event, coverage is quite low with one reporter, and rises steadily as more reporters are added to the environment. In this fashion, we can achieve a reasonable level of coverage by simply adding more and more reporters to the world, although this is not an indefinitely scalable solution - with too many reporters in the game, the players will find them increasingly obtrusive, impairing line-of-sight or possibly even blocking movement. In this experiment, the reporters ignored one another. As a result, two or more reporters often group together and commentate on the same events, which has benefits in some situations (if one reporter is killed then there is still another present to see the events), but also drawbacks (there are fewer reporters available to commentate on events elsewhere in the world).

Figure 4 illustrates the results for static role assignment, where two reporters were positioned to permanently watch over the two flag-bases, one at each, and the remaining two reporters were allowed to patrol as in the previous experiment. At first glance, it might be expected that having a reporter positioned at each flag-base should theoretically mean that 100% of flag take events are reported; however, the reporters are often killed as the flag-bases tend to be combat hot-spots, and they will also follow a player who collects the flag in an attempt to find out what happens to it. In both cases, it may take some time for the reporter to travel back to the flag-base, and during this time the flag may be taken again. The results reflect this, offering only a slight improvement over the previous data. We also tested a similar case to figure 4, where the flag-base watching reporters were not allowed to follow a player who took the flag, which gave slightly worse results - the flag take event coverage was roughly the same, but coverage of pick-ups, drops and returns were all lower.

Figure 5 shows data for dynamic role assignment with four reporters and an editor. The editor attempted to keep one reporter watching each flag-base at all times, and in the case of assigned reporters being killed or following other events it would reassign the closest idle reporter to take over. As can be seen, the flag take and flag return event coverage are both higher due to the flag-bases being under observation a greater fraction of the time, and for coverage in these two events we are getting performance from 3 reporters and 1 editor that is roughly equivalent to 4 reporters with no editor. Flag drop event coverage also sees an increase with an editor, although less than the takes and returns, possibly due to there being a flag-base watching reporter to follow any flag carrier (and thus see when they drop the flag) more often. Flag pick-up events actually have lower coverage, the reasons for which are somewhat harder to pinpoint. One possibility is that idle reporters are being reassigned to

watch flag-bases when the flag-base watchers go to follow a flag carrier. There are therefore fewer idle reporters on patrol who are likely to see flag pick-up events take place at any given moment. Flag pick-up events occur less often than all the other types of event, so it seems worthwhile to trade slightly poorer coverage of pick-ups for a greater improvement in the coverage other events.

Table 1. Reported events by role type.

	No Role	Static Role	Dynamic Role
Events Correctly Reported	2521	2788	2987
Percentage of Total Events Reported	59.3%	61.4%	69.1%

Table 1 shows the overall coverage for the same three sets of roles as in the previous charts. The results confirm our hypotheses. Overall, assigning static roles to reporters improves even coverage by a small amount (2.1%). Although the performance of reporters with dynamic role allocation is lower when detecting flag pick-up events, overall there is an improvement in coverage of events of 7.7% compared with static roles, and 9.8% compared with no role assignment. These results confirm that reporters require adaptation strategies to improve their performance. Additionally, however, they illustrate that the editor plays a key role in our system, and therefore our framework of separating out reporters, editors and presenters is a promising line of enquiry.

7. FUTURE WORK

We are aware of a number of limitations of our current implementation of our framework that need to be addressed in the short term.

Our current reporters occasionally report events incorrectly. It appears that each reporter will produce a report of an event that never happened roughly once per ten minutes, on average. They also correctly identify events, but report them as being carried out by an incorrect player with roughly the same frequency. This becomes a more serious problem as we add more reporters, since a larger number of erroneous reports are produced. In order to cut down on the number of erroneous reports produced, we are currently working on the implementation of validation and verification of reports received by the editor, for example by requiring the same report to reach an editor from multiple sources before being accepted, and perhaps discarding any reports which contradict those produced by another reporter.

Another limitation of our current reporters is that they have a no sense of self-preservation. Although Gamebots provide messages that alert reporters of when they are in danger of being hit by various incoming projectiles, these are currently ignored. Even if reporters could take avoiding action, it is inevitable that they will sustain injuries during the course of a game. It would therefore be interesting to see what happens if we allow them to disregard editor orders temporarily and seek out health packs in order to stop themselves being killed. However, this requires that editors are able to deal with uncooperative or unresponsive reporters, although they will have to do this anyway since they can occasionally become disconnected from the game server or get stuck.

There are also limitations with our current editor. It does not deal with the "cost" of reporters' deaths very elegantly, and always selects the closest idle reporter to replace one which has been killed, without taking into account other considerations. For example, a reassigned reporter may arrive at the location at which an event was occurring only to find that the event has since finished, and that there is nothing of interest left to report upon. We intend to investigate the typical duration of various events, so that an editor can compare this to the cost of reassigning a reporter and calculate whether it will arrive at the event in time.

Beyond these short-term fixes and extensions, we anticipate a number of broader future developments. We intend to create a range of presenter agents targeted at mobile phones, conventional graphical user interfaces (for example, based upon a scrolling tickertape display used to enhance group awareness in cooperative work environments [16]) and also for 3D interfaces, perhaps building upon the previous generation of animated talking head news reporters [17] to provide queued players with information about events in a game or possibly even to offer an in game news service. The latter might also involve helping to position virtual cameras within a virtual world as part of live broadcasts to external observers, perhaps during game tournaments or as part of future television shows (see [18] for an example of creating a live TV broadcast from an online game and a discussion of virtual camera techniques). We also intend to extend our approach to a broader range of games, which will require the ability to reason about a more diverse range of human activities, especially for large-scale persistent games that may involve many players, many different objects and interactions and a broader rage of social situations. Finally, we intend to apply our techniques to the emerging area of pervasive and mixed reality gaming in which games reach out into the physical world through devices such as mobile phones, enabling players to access a game while on the move, providing them with location-based experiences, and supporting games in which online players collaborate with those on the streets [19]. Commentating agents offer one way in which online games might reach out to relatively low-powered devices such as mobile phones.

8. CONCLUSION

We have proposed a multi-agent system to address the challenges involved with delivering commentary on game events to non-game participants. Our strategy involves the delineation of the functions of reporting, editing and presenting. We have discussed the merits of our system with regard to particular facets of reporting, and presented some preliminary results to justify our approach. We have examined the relationships between humans and agents in a reporting system, with particular emphasis on the privacy concerns that arise, and outlined methods of dealing with these concerns such as the embodiment of reporters. We have also examined the relationships between agents of the different proposed roles, discussing the need for co-ordination in order to provide optimal reporting coverage. Our test data helps to illustrate the benefits of some aspects of the framework, such as the positive impact an editor can produce by co-ordinating the efforts of reporters. We have also briefly examined the prospect of delivering reports to different types of device with the use of presenters. Although our framework remains to be tested across its entirety, we have populated each layer with preliminary designs and illustrated the benefits of this approach to gaming. Whilst we have chosen a particular game engine and game type with which to implement and test reporter agents, much of our prototype system is flexible enough to migrate between systems

and between game types. For example, in order to allow our prototype system to function with a 'Domination' game-type, reporters would simply need to recognise a different series of events to 'Capture the Flag'. All other agents' functions, however, remain identical. There remains much scope for further developments within this framework and we hope to realise this potential in various facets of future work

Finally, we would like to re-emphasise the strength of our approach with respect to the privacy issues surrounding dissemination of game data. Despite the inclusive nature of on-line gaming, and the situations in which such data might be presented, privacy remains an important issue for gaming communities. In embodying our approach alongside an analysis of the nature of reporting, we hope to have illustrated the importance of relationships in agent privacy. Firstly, in the relationship between humans and agents we have been sensitive to the continual requirement for subtlety in the allowing participants and players to feel comfortable with heterogeneous and large-scale data parsing and transmission. Secondly, we have illustrated how interrelated these difficulties are with privacy relationships between agents - the very coordination that occurs in the multi-agent system has direct impact on the relationships between participants and players and agents in gaming systems.

9. REFERENCES

[1] Noma, T. and Badler, N., A Virtual Human Presenter. *IJCAI Workshop on Animated Interface Agents: Making Them Intelligent*, (1997), 45—51.

[2] André, E. and Rist, T., Presenting through Performing: On the Use of Multiple Animated Characters in Knowledge-Based Presentation Systems. *International Conference on Intelligent User Interfaces*, ACM, (2000).

[3] Bates, J., Loyall, B., Reilly, W. S., Broad Agents. In *Proceedings of the AAAI Spring Symposium on Integrated Intelligent Architectures*, Stamford University, (1991).

[4] Laird, J., It Knows What You're Going To Do: Adding Anticipation to a Quakebot. *AAAI 2000 Spring Symposium on Artificial Intelligence and Interactive Entertainment*, (2000).

[5] http://www.planetunreal.com/ttm/ (verified January 2004)

[6] Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I. and Osawa, E., Robocup: The Robot World Cup Initiative. *Proceedings of the First International Conference on Autonomous Agents* (Agents'97), (1997), ACM, pp. 340—347.

[7] Frank, I., Tanaka-Ishii, K., Matsubara, H., and Osawa, E., Walkie-Talkie MIKE. *RoboCup 2001: Robot Soccer World Cup V*, (2001), Springer-Verlag, pp. 343—349.

[8] Drozd, A., Benford, S. and Fraser, M. *What's the Story? Extracting Scenes From Improvised Role-Play.* Technical Report Equator-02-031, Equator – Nottingham, 2002.

[9] Kendon, A., Spatial organization in social encounters: the F-formation system. *Conducting interaction: Patterns of behavior in focused encounters*, (1990), Cambridge University Press, Chapter 7.

[10] Logan, B., Fraser, M., Fielding, D., Benford, S., Greenhalgh, C. and Herrero, P., Keeping in Touch: Agents Reporting from Collaborative Virtual Environments. *AAAI Spring Symposium*. (2002), AAAI Press.

[11] http://unreal.epicgames.com/ (verified January 2004)

[12] http://www.everquest.com/ (verified February 2004)

[13] Adobbati, R., Marshall, A. N., Scholer, A., Tejada, S., Kaminka, G., Schaffer, S. and Sollitto, C., Gamebots: A 3D Virtual World Test-Bed For Multi-Agent Research. *Agents '01*, ACM, (2001).

[14] Sloman, A. and Poli, R., SIM_AGENT: A toolkit for exploring agent designs. *Intelligent Agents II - Proceedings of the Second International Workshop on Agent Theories, Architectures and Languages*, (1996), pp. 392-407.

[15] Sloman, A. and Logan, B., Building Cognitively Rich Agents Using the SIM_AGENT Toolkit. *Communications of the ACM, 42,* 3, pp. 71—77.

[16] Fitzpatrick, G., Parsowith, S., Segall, B., and Kaplan, S., Tickertape: Awareness in a Single Line. *CHI'98 Summary,* 281282. ACM, (1998).

[17] http://www.ananova.com/video (verified January 2004)

[18] Greenhalgh, C., Benford, S., Taylor, I., Bowers, J., Walker, G. and Wyver, J., Creating a live broadcast from a virtual environment. In *Proceedings of ACM Computer Graphics (SIGGRAPH'99)*, Los Angeles, USA, (1999), pp. 375-384.

[19] Flintham, M., Anastasi, R., Benford, S., Hemmings, T., Crabtree, A., Greenhalgh, C., Rodden, T., Tandavanitj, N, Adams, M, Row-Farr, J., Where on-line meets on-the-streets: experiences with mobile mixed reality games. In *Proceedings of CHI'03*, ACM, (2003).