Visualizing Competitive Behaviors in Multi-User Virtual Environments

Nate Hoobler* University of Virginia
Greg Humphreys† University of Virginia
Maneesh Agrawala‡ Microsoft Research

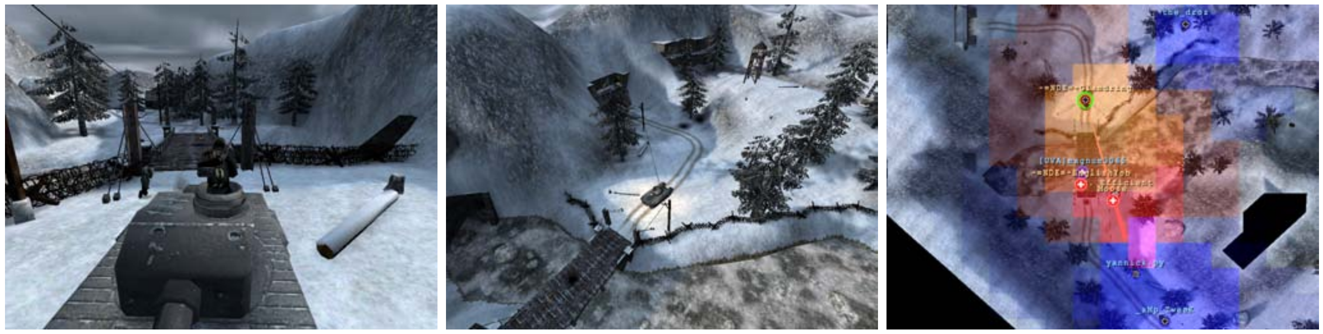

Figure 1: In first-person games, observation modes are typically restricted to an over-the-shoulder chase camera (left) or a floating-player view (center). Both views make it very difficult to understand complex team-oriented actions that have an inherent global nature. We present a novel game observation system (right) that extracts high-level semantic information about the action taking place in a game and displays it visually. By emphasizing important low-level details and overlaying them with high level action summaries, we provide a unique and insightful new view of the environment and behaviors therein. Using our system, it can now be seen that the red team is holding the bridge at the center of the map against a frontal assault by blue, but is also being flanked from the North by a lone blue player.

Abstract

We present a system for enhancing observation of user interactions in virtual environments. In particular, we focus on analyzing behavior patterns in the popular team-based first-person perspective game *Return to Castle Wolfenstein: Enemy Territory*. This game belongs to a genre characterized by two moderate-sized teams (usually 6 to 12 players each) competing over a set of objectives.

Our system allows spectators to visualize global features such as large-scale behaviors and team strategies, as opposed to the limited, local view that traditional spectating modes provide. We also add overlay visualizations of semantic information related to the action that might be important to a spectator in order to reduce the information overload that plagues traditional overview visualizations. These overlays can visualize information about abstract concepts such as player distribution over time and areas of intense combat activity, and also highlight important features like player paths, fire coverage, etc. This added information allows spectators to identify important game events more easily and reveals large-scale player behaviors that might otherwise be overlooked.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques H.5.2 [Information Interfaces and Presentation]: User Interfaces—Interaction Styles H.5.1 [Models and Principles]: User/Machine Systems—Human Factors

Keywords: Visualization, Games, Spectating

*nsh3h@cs.virginia.edu
†humper@cs.virginia.edu
‡maneesh@research.microsoft.com

IEEE Visualization 2004
October 10-15, Austin, Texas, USA
0-7803-8788-0/04/$20.00 ©2004 IEEE

1 Introduction

Observing multi-player games has practical applications beyond pure entertainment. The trend in today's games is toward more accurate simulations of reality. In particular, team-based multiplayer games have evolved from free-for-all "deathmatch" combat to subtle, goal-based, multi-agent combat simulations. Because of the coordination and tactics usually required for such games, parallels can easily be drawn to warfare training simulations, or to any other immersive situation where multiple users must cooperate, such as disaster response simulation. By improving the ability of an observer to quickly understand the overall structure of the action, these simulations can be analyzed more completely, so that the performance of individual players, teams, or strategies can easily be evaluated.

In this paper, we present Lithium, a system that enables high-level analysis of competition taking place in multi-player games. Games have steadily increased in complexity as computer and network speeds allow more individuals to interact in environments of ever-increasing realism. With each new generation, these games accumulate more players who compete on teams in official events for both recognition and prizes. Despite the growth of competitive gaming as a bona fide sport, the act of observing these games has changed very little since the genre first appeared. The core challenge for an observation system is one of managing information density. Indeed, the design of an effective observation system for a video game is very much an information visualization problem.

Currently, game spectators usually have two choices: either they can focus on a specific section of the arena and the events that occur within that limited context, or they can observe a large area in order to try to make more global evaluations. The former approach suffers because its small scope does not allow the user to see actions in a global context, which is often crucial for evaluating those actions' efficacy or importance. The latter mode is usually not much better; though the spectator can now see a physically larger area of the arena, this only increases the number of players and actions that he must keep track of. Because of this added burden on the observer, important details can easily be missed and the observer can lose track of how the overall state of the game is evolving.

To evaluate our methods, we used the game *Return to Castle Wolfenstein: Enemy Territory*, a multi-player team-based game where two groups of 6 to 12 players attempt to complete competing goals. Usually, one team tries to complete a set of objectives, while the opposing team attempts to stop or delay them until a pre-set time limit is reached. Each player belongs to a class (engineer, medic, soldier, etc.) and must work together with their teammates in coordinated actions. Team games are good candidates for information visualization, as group behaviors can be aggregated based on the dual objectives present in every match. Free-for-all games, where each player is working only for themselves, are less amenable to information aggregation since each player has a separate objective. Also, since players in team games are are, by design, cooperating to achieve pre-set goals, large-scale behavior patterns are much more likely.

Our solution to these problems is twofold. First, we iconify many semantic aspects of the action, such as munitions usage and player classes, orientations, positions, and paths. These overlays allow spectators to see this information easily from a distant vantage point, making judgments of large-scale interaction much easier. These semantic icons work well for simple and inherently spatial information such as player positions and firing patterns. However, these visualizations are not well suited for more abstract concepts such as which team has control of regions that neither team may directly occupy, what the average distribution of players is for each team, or what areas of the map may be of specific interest to the viewer. For such tasks, we provide a colored map grid overlay that aggregates weighted variables to reveal a very high-level summary of a particular aspect of the action taking place.

2 Related Work

Visualization techniques for first-person video games fall into two broad categories: spectacle visualization and analytical visualization. Spectacle visualizations are designed to present the in-game action in a form that is entertaining and exciting. Analytical visualizations are designed to aid the observer in understanding the higher-level state and overall dynamics of the game. We consider each of these visualization categories in turn.

2.1 Spectacle Visualization

Spectacle visualizations often focus on one specific player or area of combat to provide a human-level view of dramatic moments in the game. Many systems provide a phantom-player mode in which spectators can move through the environment and observe the action as a disembodied first-person “phantom player”. Some systems also provide a player-following mode in which spectators observe the progress of a specific player by viewing the game through either that player’s eyes or a camera which lags slightly behind the player. These modes often provide no more insight into the overall action than an individual player would have. In fact, in the player-following mode, the lack of direct control over the viewpoint can be extremely disorienting.

Halper et al. [5] have designed a system that automatically summarizes a game as a series of highlight clips. Their technique builds on prior work in automated camera planning [6, 7, 3] for computing effective shot compositions and video summarization [8, 1]. By focusing on only the most dramatic moments of a game, a spectator is given a condensed version of the overall match, with all of the exciting moments condensed into a short period.

The drawback of spectacle visualizations is that they focus on showing the game from a human level viewpoint and therefore only allow the observer to see a small subset of the interactions taking place within a limited region of the environment. Such visualizations can make it difficult to understand the overall context of the game because they may distort the flow of time and because they may hide many important details about how the players arrive at their current states.

2.2 Analytical Visualization

One goal of analytical visualizations is to reveal the overall dynamics of the game by showing all of the player interactions taking place in the environment. Many games provide overhead radar views showing the location of all the players as dots superimposed on a simplified map of the environment. Niederauer et al. [13] recently pointed out that this approach fails for densely occluded multi-story architectural environments because the 2D map cannot show which floor each player is located on. They solve this problem by automatically generating exploded views of the environment to separate the stories and thereby show all of the action taking place on each floor. However, revealing all the action using either a radar view or a multi-story exploded view can have the unintended effect of making it difficult for observers to decide where to focus their attention.

Analytical visualizations may also be aimed at presenting larger-scale player and team behaviors that may take place over longer periods of time. While such visualizations are rarely found in video games, mapmakers have a long history of overlaying information about the locations, actions and other properties of individuals, groups and armies on geographical maps [11, 12]. The overlays are formed via statistical aggregation techniques and are commonly used in the Geographic Information Systems (GIS) to reveal overall trends [9]. For example, to show the overall flow of movement for an entire army, the mapmaker would aggregate the positions of individual soldiers over time. Perhaps the most famous example of such a visualization is Minard’s map of Napoleon’s march on Russia [14], which simultaneously depicts the movement and attrition of the army over the course of the battle. Similarly, city planners [10] and landscape designers[4] often create maps showing the aggregated motion patterns of people traversing the environment. Recent work by Chittaro and Ieronutti [2] has also applied this technique to virtual environments as a way of dectecting distnct behaviors among the users, as well as visualizing the overall usage patterns within the test environments.

Our work is aimed at producing analytical visualizations of player interactions in first-person video games to give the observer an overall sense of the game dynamics. We combine the techniques of showing player paths on an overhead map of the complete environment in real-time with overlays depicting other forms of statistical data.

3 System Overview

In order to understand a match, we must know what actions and events are most important. We must in turn determine what measurable values and phenomena are indicative of the events, so that we may detect and display them. Ideally, these metrics should be designed such that they can be used to easily answer important questions that users may have about a given scene.

In order to select the most effective metrics for match evaluation, we decided to familiarize ourselves with the game and tactics used therein as much as possible. First, we studied the player experience by playing the game for some time, becoming familiar with its intricacies. Once we had a basic understanding of the games mechanics, we proceeded to observe a series of matches using the engine's existing spectator mode. As we observed the matches, we took notes on questions that arose and important events that occured. Based on our experience, we found that the following questions were often the most important:

"Where are players of each team concentrating?"

"How is a team organizing itself?"

"Where are players likely to run into opposition?"

"Where is conflict occuring?"

"What are the tactical details of this conflict?"

"How did the current game state come to be?"

After repeated viewing of a series of matches, we identified a set of characteristics which we felt were helpful in answering these questions. By tracking player positions and classes, we are able to present detailed visualizations of how each team is organized within the game world. Furthermore, by aggregating position information over time, we can show where each team tends to have more influence, both immediately and historically, thus predicting the areas where players are likely to run into the opposition, as well as where they have already come into contact. Aside from position, we track other traits as well, such as the players' health and score, and when shots are fired. When combined properly, these features can provide tremendous illumination about the forces which drive the game.

We present this information in two different ways. "Local" visualizations iconify elements of the action taking place at specific places in the map, such as player positions and shots being fired. These visualizations help the observer understand the situation surrounding an individual player or event. "Global" visualizations present statistical data which is aggregated over the course of the match. These visualizations help to understand more high-level trends and behaviors.

3.1 Local Visualizations

We provide four different types of local visualizations: player glyphs, player paths, tracer fire, and fields of view. Each of these visualizations reveals some high level aspect of the action.

Player Glyphs In many objective-based team games, players are divided into classes. Players from each class have different abilities and are assigned different tasks in order to complete the overall team objective. When looking at the position of players, it is often very important to understand where, for example, the engineers are located versus the medics. Figure 2 shows an example of player glyphs. Here, a blue engineer (shown outlined in green) has achieved his team's objective of advancing to the northeast turret and placing dynamite. Although there are several red team members near him, the cross glyph indicates that they are dead. The red engineers can clearly be seen clustered in the southwest corner of the map, much too far away to reach and defuse the dynamite in time. This position is surely a win for blue, a fact that would be impossible to see if the players were not iconified and classified in this way.

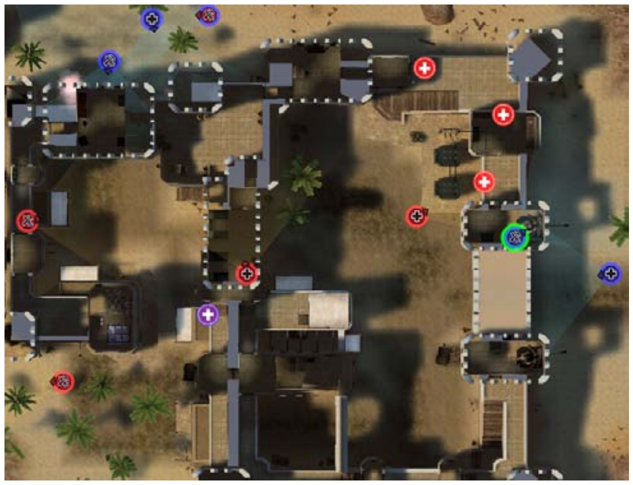

Figure 2: **Player Glyphs**. Each player's team and class is shown through color-coding and icons. Also notice that a player can be selected, allowing the observer to view information specific to that player and causing him to be outlined in green.

In addition, the team-colored halo surrounding each player can be used to encode quantitative information about that player. There are several such values that could be of interest to spectators. For example, the radius of the halo can encode the player's health. This can be important because a player with low health might behave very differently from the rest of his team; either fighting in an atypically conservative fashion or leaving the front lines in search of medical assistance. In Figure 3 we see that the red player has taken some damage, and thus his halo is much smaller than the blue player (who is at full health). This added context helps explain why the red player appears to be retreating away from the blue player towards the added defense of higher ground. Another visualization that we have found to be useful is to encode a player's "experience" in their halo. This way, an observer can see if one player is doing substantially better (or worse) than his peers, and might choose to temporarily switch to a first-person spectating mode focused on that player.

Player Paths Often, the position of players is not enough to understand why a player or team is at an advantage; we must know how their spatial configuration was achieved. We can show the recent path taken by each player, as shown in Figure 3(a). The path is thicker at more recent instants in time. In this figure, we can clearly see that the blue player turned back from his assault on the red player to hide behind the terrain. In this case, a spectator might wish to know that the blue player was in the midst of combat when he ducked for cover, and is not lying in wait for unsuspecting enemies to arrive.

In Figure 3(b), we see two engineers moving up an enbankment to build a ramp, their goal in this part of the map. By moving along very different paths, they ensure that the red team cannot adequately hold off the assault, and that the blue team will achieve its objective. Notice also in this figure that the tracer fire visualization shows that the engineers are being well supported by cover fire from further back on the beach.

Tracer Fire By showing the paths that munitions take through the game, the nature of combat is often made much more clear. The tracers are rendered as triangles which fan out from the firing player. Representing the tracers as wedges gives several benefits.

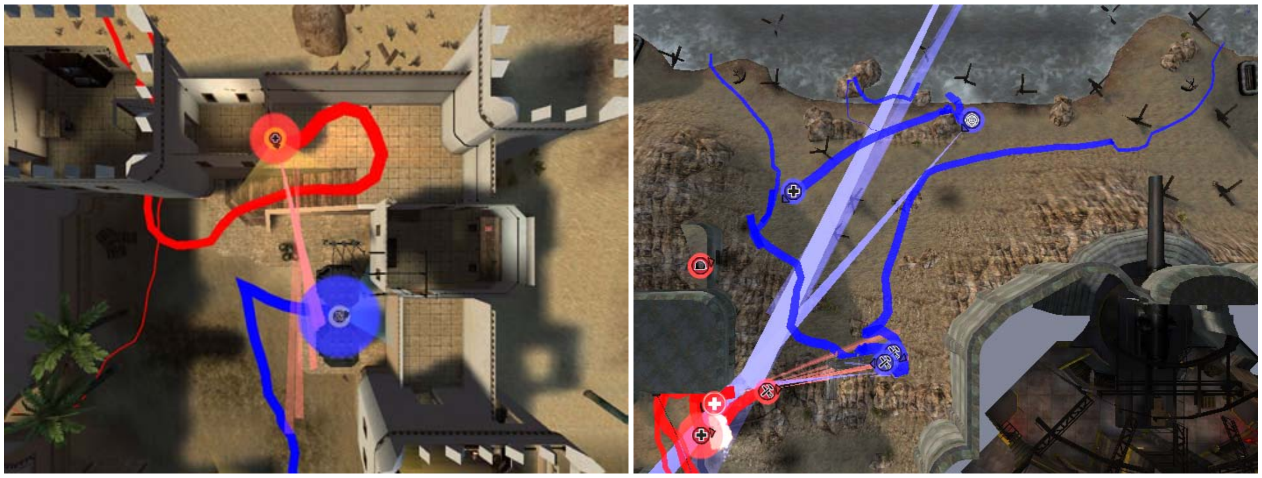
(a) ducking for cover (b) dual assault

Figure 3: **Player Paths**. The path taken by a player to their present position is shown as a line of variable thickness (thicker indicates more recent movement). This can be crucial to understand how a tactical position has come about. In (a) we can see that the blue player has ducked behind an architectural feature for cover. In (b) the separate paths taken by the two blue engineers ensure that the red team cannot adequately respond to the attack and at least one of them will successfully complete the team objective.

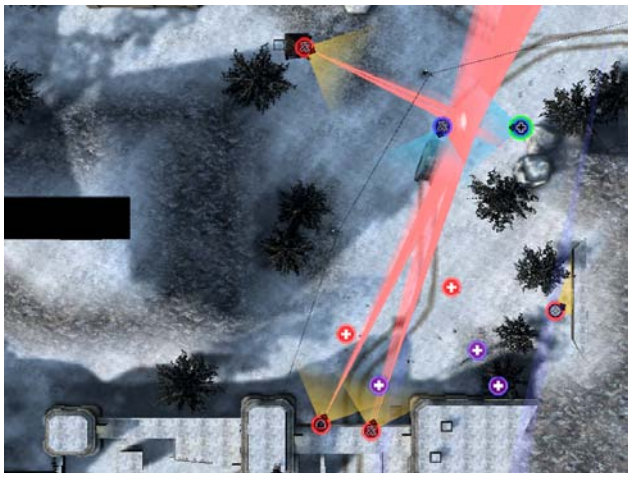
Figure 4: **Tracer Fire**. The origin of and path taken by munitions is made clear. Notice that it is now clear that the blue player is caught in crossfire, which would be difficult to see without this visualization.

First, the expansion of this wedge corresponds roughly to the accuracy the weapon at that range, thus giving the observer an idea of how effectively the player is expending his ammunition in the given context. Second, by making the tracer a wedge, we give it a concrete ”direction” of travel. This makes it much easier to determine at a glance both where fire is originating and where it is targeted. This distinction would not be nearly as easy to make if the tracers were represented with a directionless primitive. In Figure 4, we see two blue players who are caught from multiple directions in the red team’s crossfire. It is clear that red will achieve a combat victory in this scenario. Without the tracer fire, we would only see the blue players die, but would not realize that the superior positioning of the red team lead to their defeat.

Fields of View A cone of vision indicating each player’s current field of view can optionally be drawn. This cone encodes both the direction that the player is facing, as well as what portions of the map are visible to that player. Since players are more vulnurable to attacks from directions which they are not facing or do not expect, this feature can indicate possible weak points or paths of attack against a line of defenders.

3.2 Global Visualizations

Some aspects of gameplay are too global to be adequately captured by an iconification of a single action element. To deal with this issue, we divide the game arena into a *coverage map*, a two-dimensional grid of cells, each of which tracks a variety of measures of the gameplay such as the amount of combat or influence each team may be exerting within its borders. By tracking this information, we can later parse it and find important correspondences to actual in-game events, as well as determine visualization combinations that use this data to highlight certain behaviors in the overall game landscape.

To visualize the coverage map, we draw a color-coded overlay over the game’s terrain. This allows the information to be evaluated in context and more readily integrated into the overall observation experience. The color for each cell is determined by the *emphasis mode* of the map (i.e., what information the spectator has specified as important), which weights the source data fields for the cell to effectively focus attention on those gameplay features for which the spectator is looking. By changing the emphasis mode of the coverage map, the user has the flexibility to quickly and easily compare multiple game factors. In this section, we show a single situation with all of the coverage map emphasis modes we have implemented, to highlight all the different subtle gameplay aspects taking place at this instance.

Occupancy The most commonly used global visualization mode is the *occupancy map*. This mode encodes which team has

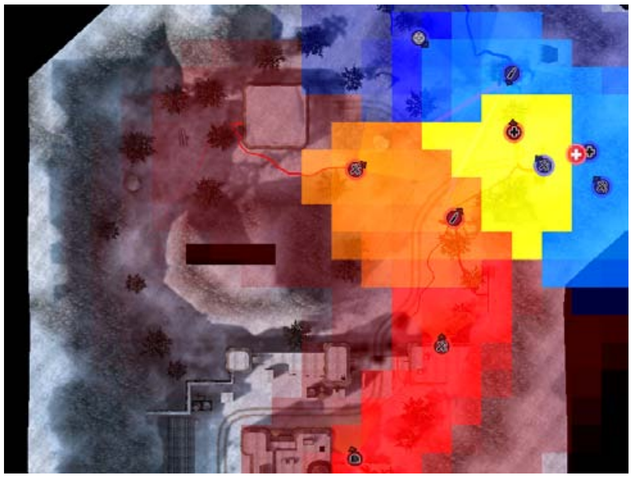

Figure 5: **Occupancy Coverage Map**. In this image, we can see that a large conflict is occuring in the northeast quarter of the map, with red reinforcements moving in from the south. Furthermore, we can see that the red team's northern flank is undefended, and that the blue player farthest to the west is well positioned to sneak around red's defenses and gain a positional advantage.

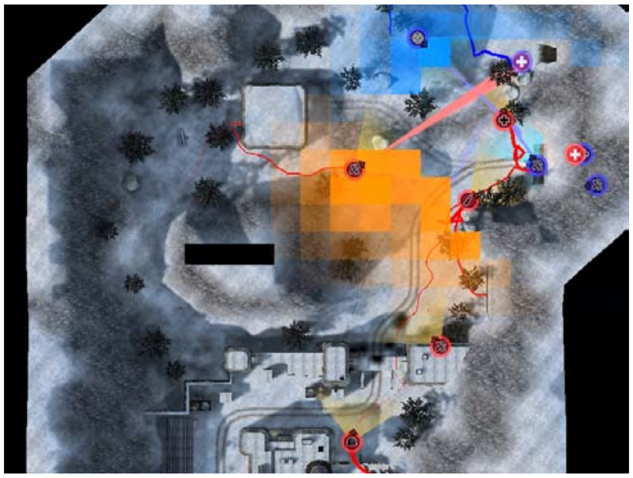

Figure 6: **Support Fire Coverage Map**. Orange and cyan cells indicate areas where red and blue players are firing weapons, but are not engaged in close combat themselves. In this image, both red and blue players are supporting their teammates who are involved in a battle in the northeast end of the map. This visualization is crucial for understanding both how well a team has set up its defenses and conversely what areas a team must attack in order to proceed.

most recently occupied a location, as shown in figure 5. The color of each cell indicates which team last controlled that cell, and the brightness of that color indicates how recent that occupation was. Each cell's color indicates which team currently exerts the most control over a region. Brighter colors (red and blue) indicate more recent occupation, while darker colors (dark red and dark blue) denote areas in which neither team currently has a strong presence, but where one has historically exhibited more control. Orange and cyan indicate where members of the red or blue team are participating in combat. Yellow cells mark contested regions where close combat is occuring.

The occupancy map gives a clear view of where the forces are currently located, where they have been recently, the overall structure of the team's position, and the location of combat. If the observer wants an overview of how a match unfolds, this can be a more effective visualization than a detailed one showing local player details.

Support Fire In team games, it is often advantageous to position a few players away from the front lines in order to provide supporting fire for the less combat-oriented players. For example, an engineer might need time to construct a bridge, and during that time he is very vulnerable to attack. In this case, a sniper might provide cover fire to prevent opposing forces from approaching the engineer and killing him without opposition. The skill of these snipers is frequently critical to the success of a team's mission. In Figure 6, we show areas where support fire is originating. Here, the red team has positioned a machine gunner on high ground, placing them at a large tactical advantage. Being able to see the evolution of support placement over time can give substantial insight into the reasons for a team's success. may or may not be the best allocation of red's forces, an insight that could prove Additionally, notice that the blue player at the top of the map would be well positioned to take out the gunner, but instead focuses his attention on the area of direct combat. This image suggests that the blue team would be better served if the sniper instead focused his attention on the red machinegunner instead of supporting the combat directly.

Medic Efficacy In free-for-all games, killed players often just respawn at various locations in the map. Many team games, however, empower players in the medic class to revive killed teammates. Obviously, the skill of a team's medics will be crucial to its success, as dead players cannot participate in the completion of the team's objectives. Our final global visualization mode, shown in Figure 7, highlights the behavior and skill of a team's medics. Areas in red show the location of killed players, while areas in green show the location of the medics (this mode only visualizes one team at a time). In this figure, the medic is well placed to respond to the recently fallen teammate. Medics that are out of position or not aware of the location of heavy combat can be a serious liability, so this visualization mode can provide a key insight into a team's performance.

4 Implementation

Since Lithium visualizes semantic details about the game being played, it must be implemented in a manner such that it has access both to important internal game data, and the underlying rendering engine. Non-invasive approaches such as those taken by Niederauer et al. [13] are attractive because they can work on any game, but they must reconstruct or infer any semantic information about the gameplay from only the geometry being drawn. Network proxy approaches are also unappealing in this situation because games frequently encrypt their network protocol to prevent cheating. Instead, we take advantage of the fact that *Enemy Territory* supports a modular, publicly available API which allows ust to have limited access both to internal data structures and to the rendering engine of the game itself.

There are some drawbacks to using a client-side module. First, there is sometimes important game state information that is not sent to spectators, as the server does not expect those clients to need it. Furthermore, we must use the narrow interface to the graphics engine to realize our visualizations. The client is typically a very simple shell that simply draws the world and draws simple text and 2D images on a "heads-up" display. Due to the expected usage which this API was designed for, ability to draw arbitrary 3D overlay in-

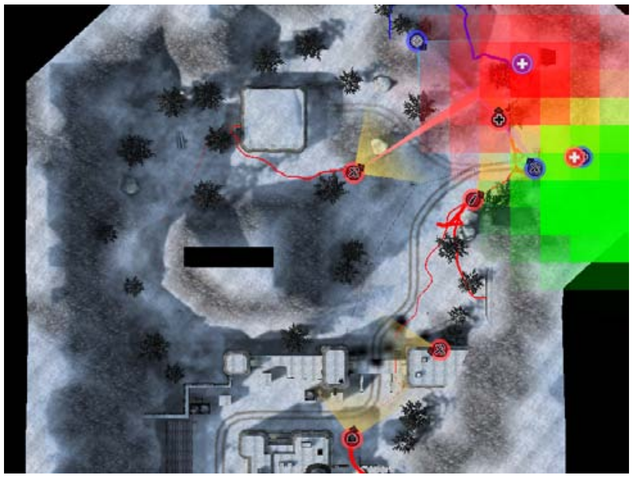

Figure 7: **Medic Efficacy Coverage Map**. In this image, the locations of dead players (red) and the medics (green) for the blue team are shown. Note that the blue medic is well placed to respond to the wounded player. If the medic were out of position, the red team could easily put blue at a material disadvantage.

formation in the world itself is somewhat restricted, limiting the complexity and detail that our system can accomodate. We do not, for example, have direct control over the projection matrix, nor are able to upload dynamic textures. Most of Lithium must be implemented using simple world space geometry or screen-aligned icons. Note, however, that these problems are not fundamental to our system; they could be easily solved if future games were designed with Lithium-style spectator modes in mind.

4.1 Glyphs and Local Visualizations

Individual players are tracked separately, keeping a history of states that are updated periodically for each player as the game progresses. Each "snapshot" of the players contains information on each one's position, score, health, activity, and other important factors. Every frame, we use the client-side graphics API to draw the local visualizations described in section 3.

Although all of the results presented in this paper show the game being visualized from an overhead orthographic view, the local visualization enhancements are actually rendered in three dimensions in the game's world coordinate system. Thus, these visualizations also function correctly in a standard perspective-based chase camera mode as well. Even in this completely local context, these overlay visualization can greatly enhance the comprehensibility of a combat segment.

4.2 Coverage Maps

Collecting data for and rendering of the global coverage maps is more complex than for the local visualizations, because decisions need to be made about what information to use and how best to present this data.

Collecting the Data The coverage map is generated from a relatively low resolution 2D grid that spans the entire game arena. Each grid cell contains several values that are estimations of the player density, medic density, medic need, and combat activity within the cell. Every frame, the entire grid is updated by iterating through all of the players and adding their influences in the different categories to the appropriate locations of the grid. Each player's influence is spread over multiple nearby cells using a kernel that falls off quickly with distance. The coverage map for the current frame is then computed as a linear interpolation between the new grid of values and the previous frame's coverage map. Blending the values in this manner has several advantages. First, it prevents values from oscillating rapidly, as would happen if the occupancy of a cell changes frequently. In fact, we carefully select the linear interpolants for each tracked variable separately, so some values more closely reflect the current game state, and others have more temporal stability.

In fact, these interpolation rates need not be constant. The rates for medic need and density, for example, vary according to the magnitude of the newly computed values. If the new density is greater than the current density or if the current density is above a given threshold, the value changes very quickly; otherwise the values change much more slowly. This way, a player entering an area will immediately have his presence correctly reflected in the surrounding cells, even if the values are extremely low at the time. However, when he leaves, the value will fall off much more slowly, causing the player's presence to "linger" in places he has recently visited, allowing the observer to see a time-lapsed history of the player's influence as he moves about the arena.

Presentation In order for the data collected in the coverage map to be useful it must be presented to the user in a meaningful and easily deciphered way. Simply blending between different colors based upon the values for each metric tracked in the cell is overly simplistic and results in unclear visualizations. The observer does not need to visualize the exact values of each metric across the grid; instead, he wants to see a summary of the status of each cell. To achieve this, we use a *modal coloring system* which, based on the values of the different metrics for that cell, present a general categorization of the cell. By making these hard distinctions, the observer can far more easily determine where borders or where important events are occuring.

An occupation coverage map, such as Figure 8 uses a cascading series of logic statements to categorize each cell, then colors them appropriately. First, it determines which team has the strongest presence in the cell by comparing the player presence metrics. If the value of the stronger team's presence is less than a specified "presence threshold" (the same value used for the linear interpolation rate), the cell contains a player's residual presence and is assigned a dark red or blue color (depending on the team). If the presence is greater than the threshold, we determine whether the cell is within overlapping regions of presence by opposing teams or whether it is within a region of combat; if so, the cell color is modified accordingly.

The "support fire" mode's coloring is not quite as complicated as the occupation coverage map. This emphasis mode defines support fire as shots fired from an area which is "uncontested" (which we define as only one team having an occupancy value above the presence threshold). By displaying uncontested areas where shots are being fired, this mode can visualize whether players who are not directly involved in a combat region are still taking part in the conflict, tying local phenomena to global behavior.

The "medic efficacy" mode is even simpler, because only one team may be monitored at a time. Initial attempts to visualize the medic densities for both teams simultaneously were difficult to decipher, because the medic needs for both teams tend to have substsantial overlap resulting from close combat. Dealing with only one team at a time, however, lets us use two simple color codes: red for medic need and green for medic presence. When these colors overlap, the

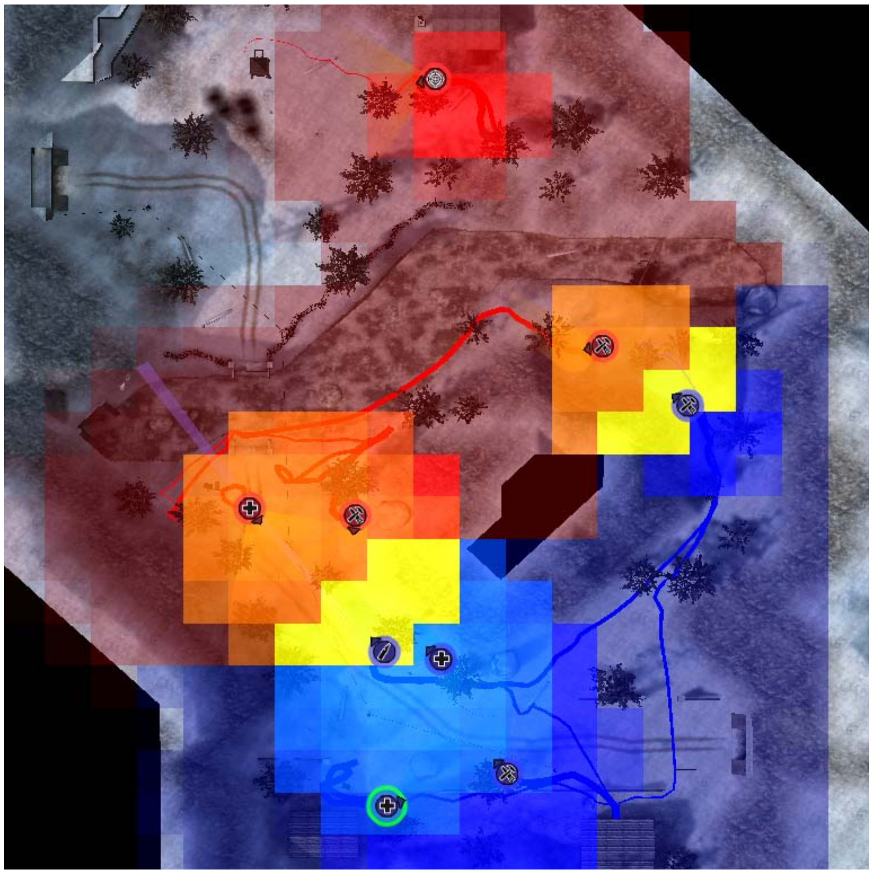

Figure 8: **Modal Coloring**. Thanks to the modal coloring system employed in the occupation coverage map, we can easily divide the game arena into several distinct areas of activity. The red team holds control of the northwest section of the map (shown in dark red), while blue pushes out from the southeast (in dark blue). Combat fronts (yellow) appear at the two chokepoints between the territories, with blue committing most of its forces to the main, southern pass. Red has also left one player in reserve to the north, to assist by providing covering fire to either front as needed.

resulting yellow cells make clear where medics are doing a good job.

5 Conclusion and Future Work

Our system is successful at presenting a visual encoding of a variety of aspects of multi-player action that make it easier for an observer to understand high level aspects of that action. Many of our visualizations reflect the evolution of an action over time, so spectators understand not only what is taking place, but how the current game state came to be. Our experience has been that these visualizations, coupled with a rudimentary understanding of the mechanics of multi-player team games, allow a novice spectator to easily discover and grasp subtle aspects of matches that even experienced players cannot find with a standard spectating interface.

In the future, we would like to objectively study the effectiveness of our system by having actual user studies. Preliminary anecdotal evidence seems to indicate that the system is quite effective in achieving its goals. However, experimentation with different presentation modes and levels of information density in a controlled environment would undoubtedbly prove fruitful.

As far as further development and features go, there are two primary directions for this work. The first is automatic detection of "interesting" or anomalous actions as they take place. Although allowing the spectator to control these parameters (as we do now) can have certain advantages, we are very interested in allowing completely passive viewing of games. A passive spectating system would automatically select visualization parameters in order to direct an observer's attention. This could also be extended to control the camera system and reposition/zoom the view to focus on important features such as a particular local action or an overall positional shift. The key advantage to automatic visualization selection is that it can be non-trivial to find which combinations of overlays and input data would most clearly show a particular feature or event in a given match. Instead of having the user manually explore this parameter space, an automation module could take a simplified set of "interest parameters" (weights that the system would use to determine the importance of player movements, combat, grouping behaviors, territory control, etc.) and automatically select which visualization configurations would show most clearly important features of the match based on the simplified weights.

We would also like to explore techniques for *analyzing* video games, rather than spectating them. The key difference is that analysis tries to answer the question of "what went wrong;" i.e., why a team succeeded or failed. For a spectator who wishes to enjoy a game, the proper interface is probably a mix of semantic overlays for explanatory reasons and through-the-eyes replays for fast-paced action. However, a team captain may wish only to understand the reasons behind his team's shortcomings, which is a different goal. For more analytical exploration of games, we would likely have the luxury of offline computation, since games can be recorded for later playback. These offline systems would map the game onto simplified models that can then be used to more easily compare different matches. One such method would be to take a set of recorded matches (all on the same arena with the same set of objectives) and monitor the positions and velocities of the players at every timestep. Once this time-based model of player density is complete for each match, the information can be merged, compared, and contrasted to find correlations or important distinctions (especially between matches that are won by different sides). This data could even be used to create a directed graph or other summary of a given arena; the resulting model would be useful in real-time analysis of other matches, allowing for quick comparison to canonical examples.

ACKNOWLEDGEMENTS

The authors would like to thank Jonathan Moses for early access to the game source code, as well as the team at Splash Damage for their help in making this project a success. Cliff Woolley stayed up late with us making sure that all our I's were dotted.

REFERENCES

[1] Matthew Brand. The inverse Hollywood problem: From video to scripts and storyboards via causal analysis. In *Proceedings 14th Conference on Artificial Intelligence*, pages 132–137, 1997.

[2] L. Chittaro and L. Ieronutti. A visual tool for tracing users' behavior in virtual environments. In *Proceedings of AVI 2004: 6th International Conference on Advanced Visual Interfaces*, pages 40–47. ACM Press, May 2004.

[3] Steven M. Drucker and David Zeltzer. Camdroid: A system for implementing intelligent camera control. In *1995 Symposium on Interactive 3D Graphics*, pages 139–144, April 1995.

[4] Stephen M. Ervin and Hope H. Hasbrouck. *Landscape Modeling: Digital Techniques for Landscape Visualization*. McGraw-Hill, 2001.

[5] Nick Halper and Maic Mausch. Action summary for computer games: Extracting action for spectator modes and summaries. In Wan Hak Man Loo Wai Sing and Wong Wai, editors, *Proceedings of 2nd International Conference on Application and Development of Computer Games*, pages 124–132, 2003.

[6] Nicolas Halper, Ralf Helbing, and Thomas Strothotte. A camera engine for computer games: Managing the trade-off between constraint satisfaction and frame coherence. *Computer Graphics Forum*, 20(3):174–183, 2001.

[7] Nicolas Halper and Patrick Olivier. CAMPLAN: A camera planning agent. In *Proceedings 2000 AAAI Spring Symposium Series on Smart Graphics*, pages 92–100, March 2000.

[8] Liwei He, Elizabeth Sanocki, Anoop Gupta, and Jonathan Grudin. Auto-summarization of audio-video presentations. In *ACM Multimedia '99*, pages 489–498, 1999.

[9] George B. Korte. *The GIS Book*. OnWord Press, 2000.

[10] Kevin Lynch. *The Image of the City*. Cambridge, Massachusetts: The MIT Press, 1960.

[11] Alan M. MacEachren. *How Maps Work*. The Guilford Press, 1995.

[12] Mark Monmonier. *Mapping It Out*. The University of Chicago Press, 1995.

[13] Christopher Niederauer, Mike Houston, Maneesh Agrawala, and Greg Humphreys. Non-invasive interactive visualization of dynamic architectural environments. In *Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 55–58. ACM Press, 2003.

[14] Edward Tufte. *The Visual Display of Quantitative Information*. Connecticut: Graphics Press, 1990.