Pacific Graphics (2014)
J. Keyser, Y. J. Kim, and P. Wonka (Editors)



# Visual Analysis of FPS Gameplay Data: From Game Design to Player Behavior

Quan Li[†1] and Huamin Qu[2‡]

[1]Online Games Division, NetEase, Inc.
[2]Department of Computer Science and Engineering, Hong Kong University of Science and Technology

**Abstract**

*Gameplay data analysis has already become an important method for analyzing player behavior in games. Visualization is a promising way to explore and gain insight into the data. In this paper, we work closely with the game designers and user experience engineers to develop a visual analytic system to help them explore the gameplay data for a novel FPS (First-Person Shooter) game specific in the mainland China. We first come up with task specifications for such a system. After that, we propose a set of design goals for our system. VisFPS, is thus developed iteratively through a complete use-centered design process. The system is divided into two parts: Macro-View to deal with the overall gameplay data to discover patterns, and Micro-View to focus on a specific game match to recreate the game scene and use it to study player behavior and verify the game design intent.*

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques I.6.3 [Simulation and Modeling]: Application—Gameplay data

## 1. Introduction

Data is heavily relied on to improve product innovation. The game industry is no exception. An important form of feedback is the gameplay data, which offers a great opportunity to analyze player behavior, and thereby improving the game design. A lot of basic statistical methods have been used to analyze the data in different aspects. However, most of them usually require certain assumptions about the data (e.g. the programming mechanism is correct, or player behavior is in accord with expectations), but gameplay data, sometimes is hard to anticipate beforehand. Also, there are obstacles for designers to adopt general recommendations to their specific design. When we interviewed with them, they found it was hard to revise their game design according to the general statistical information, as it is too vague without any guidance on which parts are good and which should be revised. Meanwhile, the amount of data is beyond their capability. Fortunately, visualization, which allows for a more explorative data analysis, is a promising tool.

In this paper, we collaborated with two game designers and three user experience experts to iteratively design VisFPS, a visual analytic system to help them understand player behavior and improve game design in the future. The log data we use does not cover all aspects of player behavior and movement details. They only contain the coordinates of players when there is a specific event. This mechanism is designed to avoid the physical transfer limitations and ensure the game fluency. The second challenge is to educate the experts about the data and the potential ways to visualize them, as none of them have much idea of visualization knowledge. Therefore, we worked closely with them and helped them explore the gameplay data in different aspects to see how they can benefit from the data. After that, we abstracted the tasks in the problem domain and proposed the task-based design rationale accordingly. Our work has two major contributions. Firstly, we reviewed the previous work, abstracted the tasks together with the front-line colleagues for the analysis of gameplay data, and proposed the corresponding design requirement. Secondly, we implemented VisFPS to help them understand game player behavior and conducted several tasks and gained new insight into player behavior.

† whuisslq@163.com
‡ huamin@cse.ust.hk



DOI: 10.2312/pgs.20141263

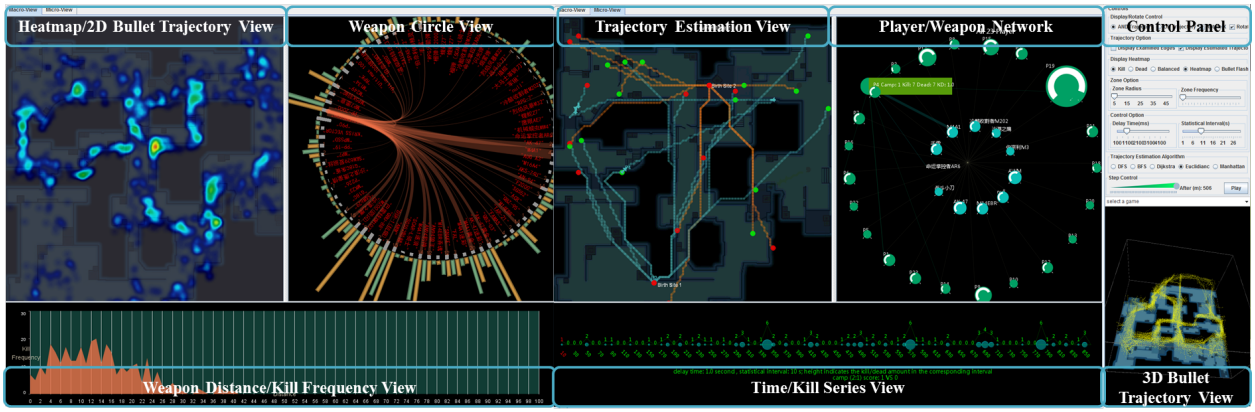


**Figure 1:** *VisFPS contains two main views: the Macro-View (the Heatmap/2D/3D Bullet Trajectory View, the Weapon Circle View and the Weapon Distance/Frequency View), the Micro-View (the Trajectory Estimation View, the Player/Weapon Network and the Time/Kill Series View (rendering the amount of "kill" events along with the time)), and the Control Panel. Each view is closely linked and targets on some specific analysis tasks.*

## 2. Related Work

Visualization techniques for FPS fall into two broad categories: spatial/temporal analysis and trajectory analysis [HM03] [CJBY08] [HHA04] [CRI06]. Perhaps the most widely used visualization is heatmap, such as Half Life: Episode 2 [Val12a] and Counter-Strike: Global Offensive [Val12b]. Another component of data from games is the temporal dimension of the data [MC09] [TKC08]. Analyzing trajectories is currently used to locate illegal bot programs in online games, examine group behavior, study player tactics, asset use, etc [Dan12]. In other games such as soccers, visualization is used to tell stories about a soccer match [PVF13]. As in real life there is no guarantee to capture anything a player does. The (*location*, *time*) data capturing the motion of moving objects is subject to uncertainty for a variey of reasons, at every stage of its generation. Two broad categories of location uncertainty models are identified [LWG*09]: pdf-based models and shaped-based models. While various works have been done, most of them focus on macro aspects. However, all of these provide us with the motivation for this work.

## 3. Problem Characterization

The time range of gameplay data we used covers a week, recording over one hundred matches for each type of game map. It records two kinds of events: "kill" and "dead", whereas "kill" represents the place where the player kills others, and "dead" represents the place where the player being killed by others. "Kill" events are associated with "dead" events, therefore each record represents a "kill" event and a "dead" event. Our task analysis is based on a research on existing work of analyzing the gameplay data in our corporation. We collected potential tasks they may be interested in. We finalized the list of analysis tasks that VisFPS needs to support. (a) **[T.1] What is the distribution of "kill"/"dead"?** Through these information, our designers can gain possible insight into the way our players are experiencing the game. (b) **[T.2] Are all game maps in accord with the designers' intent?** (c) **[T.3] Has the design intent been met in terms of different weapon usage and their performance?** It is thus possible to evaluate the impact of different weapons on different regions of a game map. (d) **[T.4] Is there any relationship between the trajectories and the performance of players?** Combining trajectories with event data is useful because paths alone do not necessarily tell us why players navigated in a particular way. The five experts are curious about cooperative behavior that may exist in our data. According to the tasks, design requirements are then presented as follows: (a) **Simple/Intuitive Design**: users prefer simple visualizations so that they can quickly understand the underlying stories. (b) **Time-embedded/Reproduction Design**: when we first presented the visualization of log data to the experts, all of them pointed out that in most cases, they couldn't understand the patterns found in the log data alone without time-embedded and match reproduction design that they can adjust by themselves. (c) **Multi-level/Linked Exploration**: it is important to understand player behavior in different scales. Thus, we provide these analysis tasks with a step by step explorative visualization that users can gradually approach to the potential patterns.

## 4. Task-based Visualization Design

The interface of VisFPS (Figure 1) consists of two main views which form a complete system that allows analysts to analyze gameplay data comprehensively. The explorative

process starts from the Macro-View, in which we will resolve the tasks (T.1, T.2, T.3). Then by selecting a particular game, the Micro-View is applied so that users can penetrate to study player behavior in a more detailed scale (T.4).

### 4.1. Heatmap/Bullet Trajectory View

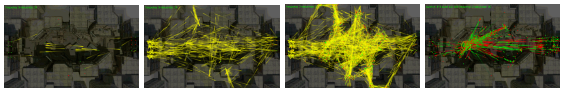

**Figure 2:** *Bullet trajectories with threshold of* 9*(AND) (first) and* 9*(OR) (second); all trajectories are represented by yellow moving arrows initially (third); by selecting an area to check up area security equilibrium, the trajectories of bullets fired into this region are represented by green moving arrows while those fired from this region are represented by red moving arrows (fourth).*

Let's start with a basic "kill"/"death" heatmap (T.1). Each kill is binned based on the kills'/victims' location and the number of records in each bin is measured (Figure 3). However, in most cases, the shooting and hitting points are usually processed separately into two individual heatmaps. The trajectories about how the bullets are fired from the shooting places to the hitting places are lost. Therefore, we encode pairwise "kill"/"dead" points into green and red dots and represent trajectories by using moving yellow arrows starting from the "kill" places to the "dead" places. One usage is to balance the map area security equilibrium (T.2). We enable user to focus on a specific area on the map by moving a circle with a default radius, as shown in Figure 2. We ignore the bullets inside the area and only calculate the bullets that are fired from or hit within this area. By wandering on the map, we can check whether the number of bullets that fire from this area equals or is close to the number of bullets that hit within this area. If this difference is significant, an unbalanced design may happen here.

**Figure 3:** *"kill" heatmap (first); 2D bullet trajectory view with selection in one AOI (Area of Interest) (second); "dead" heatmap (third) and 2D bullet trajectory view with frequency threshold of* 5 *(OR) (fourth).*

When a massive dataset need to be rendered, it is very difficult to observe any specific trajectory behavior. Therefore, we save the bullet frequency on each pixel and control the frequency threshold to display those trajectories with higher frequencies. There are two options: "AND" (trajectories of which the frequencies of the starting and ending places both exceed the threshold) and "OR" (trajectories of which the frequencies of the starting or the ending places exceed the threshold). After reducing clutter, several unusual phenomena can be observed, such as on-hook behaviors, remotely symmetrical firing and shooting, and the phenomenon that in the gaming time, the opponents lurk into the birth place of another campaign and fire, as shown in Figure 2.

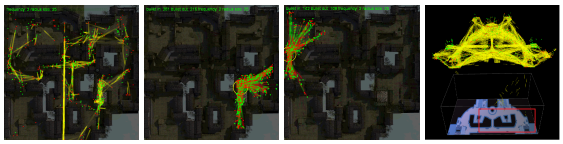

**Figure 4:** *2D bullet trajectory view with threshold of* 2 *(AND) (first); two AOIs (second and third), and 3D view to detect bugs (third) (Some players are born under the floor of the map (T.2), as indicated by the red rectangle).*

How to analyze opponent behavior tracks is another problem we care about. In Figure 3, we first show the "kill" heatmap and then select the most concentrated area in the 2D bullet trajectory view and find out that there are all together 374 bullets fired into this area but only 164 out from this area. This area is not that safe and players should pay attention to defend bullets from remote shooting sniper rifles that lie at the buttom. In the "dead" heatmap, we find several most "death-concentrated" areas. We raise the frequency threshold, display those trajectories (OR) and find three most frequent "dead" areas on the game map, as indicated in the rightmost image by red circles in Figure 3, which are all due to long-time on-hook (only for achieving "Everyday Task" awards, some players will lurk and keep still, waiting for being killed in order to quickly finish the match, resulting in many "dead" events). Furthermore, we increase the threshold to 2 (AND), as shown in the leftmost image of Figure 4. Then, we check up on two specially designed channels on the map, as indicated by the following images. Both the areas are not secure. Combined with user experiences, we conclude that when the enemies gather together in the vicinity of these two regions, players should primarily consider using thrown weapons, such as grenades or smoke bombs, etc., which could cause collective harm to enemies. In addition, when moving from places with a narrow vision into a place with a wider view through a portal or a channel, players in a team should move fast. Although we dynamically render the "kill" and "dead" bullet trajectories on the game map vividly, we lose the spatial information reflected by the height information, which is also the limitation of the projection of 3D data into a 2D plane. For example, if a map contains a bridge, the flattening will distort the values. The 3D bullet trajectories visualization would help more to alleviate the errors that can occur when data from multiple height-axis levels (e.g. a

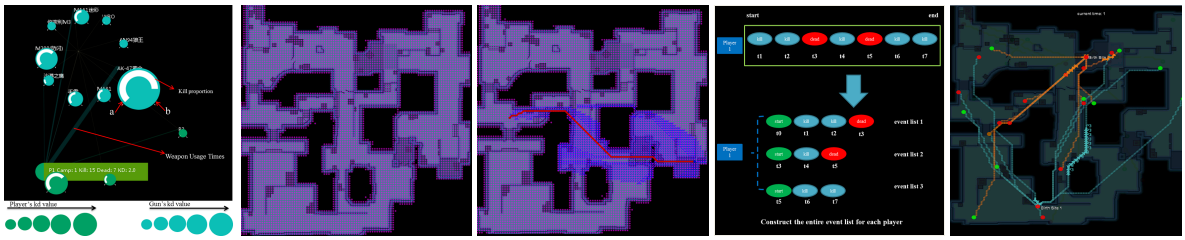


**Figure 5:** *Player/Weapon network (first); all generated node (second); estimating the path between the "start" and "end" point (third); processing steps (fourth); estimated trajectories at the first timestep in a match (fifth).*

building with two floors) are layered on top of each other, as shown in the rightmost image of Figure 4.

### 4.2. Circle View

To better understand the usage scenario of different weapons, we design the Circle View, linked with the Weapon Distance/Frequency View to visualize different performance and properties of weapons, as well as their relationships. By clicking on an arc (weapon), we can witness its connections with other weapons and the distance/frequency distribution of the selected weapon, in which the horizontal axis means different distances firing from the start to the end by using that weapon, and the height of the vertical axis represents how frequent that distance range occurs. To evaluate weapons performance and their usage scenario, the experts carried out coordinated analysis [T.3]. When they loaded a set of data gathered from a day's play-testing, they quickly got an overview of all the used weapons and their relations with others. They selected the typical rifles (*AK-47*), of which the coverage is in the range of 0 to 30. This is in accordance with the general knowledge about *AK-47*, a type of rifle suitable for short-distance shooting. On the contrary, sniper rifles, such as *AWM*, etc., generally need linear space over a long distance, so most of their firing location are on both ends of a long tunnel. The utilization rate of the Assault Rifles, including the *AK-47*, is much higher than the sniper rifles, basically firing everywhere. By picking out the thrown weapon of *grenades*, they can get an idea of when and where the players choose to use. The most common locations are those areas with obstacles blocking out players' sights, such as the corners, or the channels. The coverage is a bit further compared with Rifles, thus they can prevent from being attacked from a relatively long distance and easily cause collective harm, as shown in Figure 6.

### 4.3. Player/Weapon Network

The user experience engineers care about the potential relationship between the trajectories and their skill performance, which is measured by the ratio of "kill" and "dead" happened to him/her, called *kd* for short. We adopt the Player/Weapon Network which is composed of "Player" nodes and "Weapon" nodes. The size of each node represents its *kd* value (for a "Player" node, the radius is the ratio of "kill" amount and "dead" amount; for a "Weapon" node, the radius is the ratio of how many times the weapon is used to kill to the times the weapon held by the killed player). The proportion of the white arc inside each node represents the kill amounts to the total amount(sum of "kill" and "dead" numbers). When selecting a node, a tooltip will be shown, indicating the detailed information, and also the relations between it and other nodes (the thickness of the lines represents how many times this player kills others or uses weapons), as shown in the first picture of Figure 5.

### 4.4. Trajectory Estimation View

Combining trajectories with the temporal dimension would add a dynamic quality and allows for a better understanding of the flow of gameplay. However, in most cases, the position of players is not enough to understand why a player or team is at an advantage. Therefore, we have to estimate the position of players at an arbitrary point of time and simulate the game situation at that time. Thus, for every player, we generate a list of "kill" and "dead" events. This list would end when the final event type is "dead", which means this player has been killed by others and has to be reborn at the birth site of the corresponding camp he/she belongs to. Then we fulfill them by adding the birth site of his/her camp to indicate the start position. Therefore, the event list becomes: [$t_0$ : "start" : player id : camp : position (birth site of his/her camp), $t_1$ : event type (kill) : player id : camp : position, ... , $t_n$ : event type (dead) : player id : camp : position], as indicated in the fourth figure of Figure 5. At the very beginning of each player, we assume the triggering time of the event "start" is the begin time of the whole game match, while for others, the triggering time of the event "start" is the last event the player experiences (the triggering time of his/her last death), as the default strategy setting of the game is "0 second rebirth".

Then we estimate the trajectories of all players based on these waypoints ("start"/"kill"/"dead" event). In real games, the paths in the background map are sheltered by buildings, trees, etc. and we cannot easily detect those accessible ar-

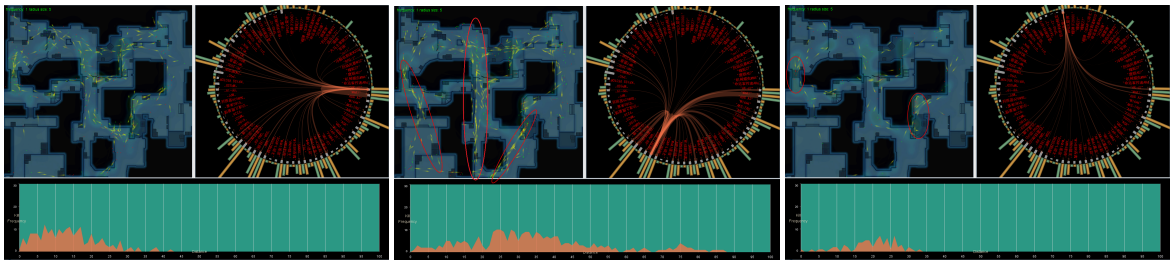

**Figure 6:** *AK-47, nearly fires everywhere and the shooting distance is relatively short (first); AWM is generally used in a linear space over a long distance (second); Grenades are adopted in areas with obstacles blocking out players' sights (third).*

eas simply from the image level, thus we use a consideriable amount of accurate historical position data to find out all the accessible regions that players can arrive at. Then we plot them together with the game map and transfer it to a bi-color image, using black color to indicate those inaccessible regions and regenerate the game map. By this way, we successfully transfer the path finding problem from the image level to the graph level by constructing the game map nodes and edges. The next step is to estimate how a particular player starts from the "start" point to the place where the "dead" event occurs. There are several algorithms in regard to how to find a path from the origin to the destination, such as the Breadth First Search (BFS), the Depth First Search (DFS), the Dijkstra algorithm and those variations based on A-Star algorithm. We customize the calculated paths by comparing the moving speed $V_a$ with the estimative moving speed in real game $V_e$. The $V_e$ is in the range of two extremums: $V_1$ (with no weapon or knife in the hand) and $V_2$ (with a particular weapon in the hand). If the comparison indicates that the calculated path by the algorithm is far impossible from the potential real trajectories of players, we will report this specific trajectory to verify its causes and use other alternative estimative paths. After comparing the generation of each algorithm, we finally adopt the A-Star algorithm.

The Micro-View (Player/Weapon network and Trajectory Estimation View) allows users to penetrate into a specific game in detail. Initially, when the gameplay data of a match is loaded, the users can quickly get an overview of all the involved players and weapons. Then, they are curious about the trajectories of excellent players and how the Standard Arm is used. They keep tracing the dynamic trajectories by using the Estimated Trajectory View. At the timestep of 56, *P1* kills *P2* while at the same time, *P2* kills *P1*, too. In other words, they all perish together. As the estimation algorithm stated, when the players are dead, they will reborn at their birth sites. Therefore, *P2* starts from Birth Site 1 and at the timestep of 72, *P2* kills *P3* by using *M200*. Meanwhile, *P1* is on the way. Six seconds later, *P1* again, kills *P2* by using his/her favorite *AWM*, while at that time, *P2* is helding a *knife* in the hand, which makes him/her move fast rather than holding a heavy weapon. *P1* keeps moving, and kills *P4* in

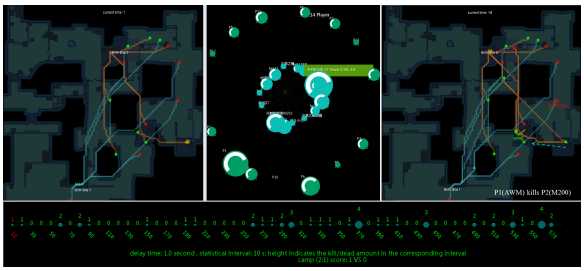

**Figure 7:** *(Above) the first timestep of the selected game (P1 firstly kills the enemy) (first); the Player/Weapon network (second) and the second "kill" event happens (P1 kills P2 by useing AWM) (third). Below is the Time/Kill Series View. Player 1 and Player 6 perform the best and they are in the same camp, which has a direct correlation with the result of the game. The snipe weapon* AWM*, and the Standard Arm* M202 *have been greatly used.*

a remote distance at the timestep of 85. About one minute later, P1 kills *P5* by using *AWM*, and continually kills *P4* nearby at the timestep of 199. *P4* is reborn at Birth Site 2, but a few seconds later, *P4* was again killed by *P1*, who is very good at using *AWM* at the timestep of 262. Until now, *P1* has killed six enemies in succession. Unfortunately, at the timestep of 294, *P1* is killed by *P2*. What about the effect by using the Standard Arm? Adjust the timestep to 365, we find that *P4* firstly uses the Standard Arm (*M202*) and kills *P3* and just one second later, *P4* kills another two enemies (*P7* and *P1*) at the same time by also using *M202*, which is quite in accord with the game designers' expectation. *M202* is just designed to cause collective harm to the enemies by killing more than one players at the same time, which is indicated in the last picture of Figure 8.

## 5. Conclusion and Future Work

We took the five experts through all the phases to help us design and validate the usefulness of VisFPS. In general, they all appreciated that it offers them an intuitive way to under-

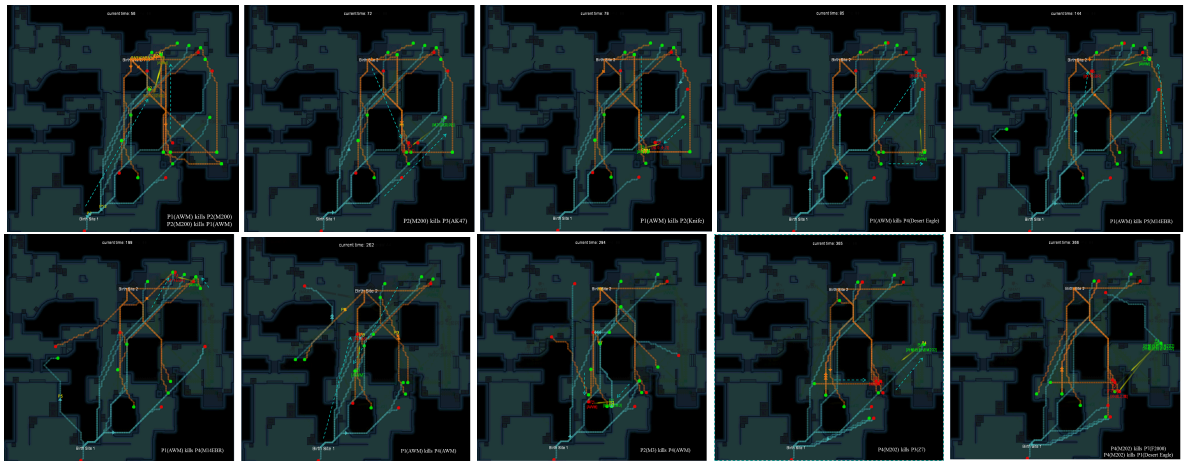

**Figure 8:** *Timestep screenshots for temporal pattern analysis, tracking on "kill" events: P1 kills P2(P2 kills P1); P2 kills P3; P1 kills P3; P1 kills P4; P1 kills P5; P1 kills P4; P1 kills P4 again; P2 kills P4; P4 kills P3 and P4 kills P7 and P1. Green points mean the "kill" events occured, whilst red points present the "dead" events.*

stand player behavior: " *I used to design a map based on the experiences in my mind, now VisFPS gives me a new perception to perceit the potential design defection in advance before beta testing*". They admitted that VisFPS offers them an innovative method to examine data. And they are more confident to verify whether the effect has met the intent of designers: "*I can see more details by using the micro view.*". Still, there are several limitations. First, the estimation algorithm works well for the simple game map, while for those much more complicated ones, it requries more efforts. Second, automatic analysis should be further developed. If the amount of gameplay data to be analyzed increases, they may feel frustrated. To deal with the first limitation, one pratical way is to add more players' behaviors and make the estimation more precise. As for the second problem, we need to apply trajectory clustering algorithms and other metrics to intelligently analyze player behavior.

In conclusion, we have presented VisFPS, a visual analytical system on visualizing and analyzing the FPS gameplay data in real-life business cases to help game experts understand player behavior. The task-based analysis and the feedback have confirmed the usefulness and effectiveness of the system. VisFPS has been applied to the Department of Game User Experience and Game Product Operation of NetEase, Inc., China' s leading Internet technology company to monitor unusual player behavior and give feedback to game designers. In the future, our work can be extended to create a comprehensive visualization framework by integrating the analysis modules for other types of behavior data.

## References


[CJBY08] CHEONG Y.-G., JHALA A., BAE B.-C., YOUNG R. M.: Automatically generating summary visualizations from game logs. In *AIIDE* (2008). 2

[CRI06] CHITTARO L., RANON R., IERONUTTI L.: Vu-flow: A visualization tool for analyzing navigation in virtual environments. *Visualization and Computer Graphics, IEEE Transactions on 12*, 6 (2006), 1475–1485. 2

[Dan12] DANKO J.: *Game telemetry with playtest DNA on Assassin's Creed.http://engineroom.ubi.com/game-telemetry-with-playtest-dna-on-assassins-creed-part-3/*, 2012. 2

[HHA04] HOOBLER N., HUMPHREYS G., AGRAWALA M.: Visualizing competitive behaviors in multi-user virtual environments. In *Proceedings of the conference on Visualization'04* (2004), IEEE Computer Society, pp. 163–170. 2

[HM03] HALPER N., MASUCH M.: Action summary for computer games: Extracting action for spectator modes and summaries. In *Proceedings of 2nd International Conference on Application and Development of Computer Games* (2003), Citeseer, pp. 124–132. 2

[LWG*09] LANGE R., WEINSCHROTT H., GEIGER L., BLESSING A., DÜRR F., ROTHERMEL K., SCHÜTZE H.: On a generic uncertainty model for position information. In *Quality of Context*. Springer, 2009, pp. 76–87. 2

[MC09] MILLER J. L., CROWCROFT J.: Avatar movement in world of warcraft battlegrounds. In *Proceedings of the 8th annual workshop on Network and systems support for games* (2009), IEEE Press, p. 1. 2

[PVF13] PERIN C., VUILLEMOT R., FEKETE J.-D.: Soccerstories: A kick-off for visual soccer analysis. *Visualization and Computer Graphics, IEEE Transactions on 19*, 12 (2013), 2506–2515. 2

[TKC08] THAWONMAS R., KASHIFUJI Y., CHEN K.-T.: Detection of mmorpg bots based on behavior analysis. In *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology* (2008), ACM, pp. 91–94. 2

[Val12a] VALVE CORPORATION: *Half-life 2: Episode two stats*, 2012. 2

[Val12b] VALVE CORPORATION: *The science of counter strike: Global offensive*, 2012. 2