How's my driving? Modeling player navigation in the Games United League of BZFlag

John C. Paolillo
paolillo@indiana.edu
School of Informatics and Computing, Indiana University

Abstract

3D game environments require players to adapt to the environment and user interfaces, imposing psycho-motor, cognitive and strategic challenges, which players may adapt to in different ways. Navigation in a 3D game is therefore complex, so that any specific instance of player movement may involve features of the virtual space, game-world physics, user-interface controls, the player's goals and intentions, and responses to other players' actions. Moreover, different players or different types of players may have different approaches to navigation. To fully understand the skills players acquire for successful game play, we need a statistical model for navigation. This paper presents one approach to modeling navigation in which players' actions govern transitions in a network of driving states. The structure of this model is explained, and exemplified from data drawn from 2421 half-hour long official matches from the Games United League of BZFlag, an online multiplayer tank battle game. Extensions of the model to obstacle or threat avoidance, strategic goal attainment and individual navigation styles are also discussed.

Keywords: Virtual space, navigation, statistical model, Markov model, state transition

Introduction

Multiplayer online games (MOGs) are a recent but highly popular cultural form used for both recreation and learning (Barab et al., 2005; Bardzell et al., 2012; Bruckman, 1995). MOGs also draw scholarly attention for what they illustrate about aesthetic experience (Bardzell et al., 2012), economics and computer system security (Castronova, 2005; Ahmad, et al., 2009, 2010), sport (Taylor, 2012), and culture and community formation (Cherny, 1999; Herring et al., 2009; Paolillo and Kutz, 2008; Ducheneaut and Moore, 2004). When MOGs employ 3D graphical worlds, navigation is central to a player's tasks. Navigation is related to aesthetic experience, e.g., as flying in second life (Boellstorff, 2010) permits experiencing graceful movement or scenic views. Navigation is also an important cost incurred in gameplay, requiring real-world time, and so bearing on the real-world value of, e.g. "gold" in Everquest or World of Warcraft (Castronova, 2005; Ahmad, et al., 2009, 2010). Quests also take time and resources, and players accomplish them by traveling to one or more (possibly distant) places, retrieving artifacts and bringing them to some other location, while navigating other hazards. Game physics, interface controls and the design of the 3D world contribute additional challenges. Different map layouts provide different affordances for game goals, and players have to learn these features and how to navigate with and around them. Different players may also adopt different approaches to solving some of the same problems, which can become routinized and later recognized as player-specific "play styles".

How then do players navigate a MOG? What are the roles of the user interface, the virtual environment and the game physics in shaping a player's navigation? What different approaches to navigation are there and how do they stand in comparison to one another? What methods can we use to uncover the nature of navigation in a 3D game?

This paper contributes to a larger program of research (Herring et al., 2009; Paolillo and Kutz, 2008; Paolillo, 2012) by examining movement within a MOG, specifically, in a corpus of saved games from the Games United League of BZFlag, a cross-platform, open source MOG in which players pilot tank avatars in a competitive, capture-the-flag game. Player position/velocity update records are analyzed to reveal the players' systematic driving choices, which are used in aggregate to construct a Markovian state-transition model of driving behavior. This model can be extended with effects for modeling individual players' navigation style, object collisions, threat changes and player interaction. This approach has

implications for modeling play style more generally that go beyond those of other current approaches to avatar movement (Liang et al., 2009).

This paper is organized as follows. We begin with a discussion of BZFlag and the Games United League, explaining features of both that pertain to avatar movement. We then consider the organization and processing of the data for the state-transition analysis of player navigation. Following this, we explore the nature of the state-transition network model, and develop a statistical model to verify the observations about the state-transition network. Finally, we discuss general conclusions form this work.

BZFlag and the Games United League

BZFlag

BZFLag is a 3D tank-battle game in which players pilot individual tanks, shoot other players' tanks and carry flags that afford special abilities ("superflags" for increased speed, shorter shot reloading, etc.) or support game goals (e.g., capturing an enemy team's flag). BZFlag originated as a LAN application written in 1992 by Chris Schoeneman for Silicon Graphics workstations, and has since become an open source project under the LGPL license, headed by Tim Riker. It has been adapted to Internet-based play and is available for most common platforms. Its current version at the time of this writing is version 2.4; the source code is written in C++ and housed on SourceForge. An ad hoc network of developers, server administrators, map designers, in-game administrators and players provide the server infrastructure and player base of the game. Active servers are listed on a central server list, and various websites and IRC channels support out-of-game communication. As open source, BZFlag is relatively easy to modify as needed for conducting analyses of gameplay.

Driving in BZFlag

Player navigation is possibly the most fundamental skill in the game BZFlag; without having learned it, and learned it well, it is impossible to achieve game goals, to share in teamwork or achieve any of the other things that the community values. BZFlag offers players a first-person perspective on the 3D virtual world through the "heads-up-display" (HUD), which comprises the main screen visible in the game client (Figure 1), and presents the information required for gameplay. The first-person field of view fills the HUD, and a number of interface elements are superimposed on top of it. A score list for connected players appears on the upper left, and a radar screen on the lower left. Chat and system messages occupy the bottom, and the center contains the "mouse-box", which is used for movement and shows the aiming of the tank. These features of the display can be turned on and off or resized as desired, using the options menus in the game client.

Figure 1. The heads-up-display in BZFlag.

Avatar movement is accomplished by positioning the mouse pointer of the mouse with respect to the two concentric square boxes of the mousebox. If the pointer is within the center-most box ("mouse centered"), the tank is stationary. If the mouse is above the center of the box and inside the outer box, the tank drives forward at a velocity proportional to the distance between the center box and the top edge. If the pointer is below the center box, the same is true, but in the backward direction, and with the condition that maximum backward velocity is half that of maximum forward velocity. Positioning the mouse to the left of the center box turns the tank left, and to the right of the box turns it right, again at a rate proportional to the distance between pointer, and the center box and the edge. Since movement velocities are limited at their maximums, full forward, backward and turning velocities are easily achieved by positioning the mouse outside the mousebox. This simple interface prohibits certain kinds of movement (e.g. "strafing", or sideways movement), but it makes controlling the tank a relatively simple action, whose elegance is highly valued by the leaders of the development team.[1] Shooting

[1] In the BZFlag forums, players often request modifications

is accomplished by clicking the (left) mouse button; normally there are a limited number of shots that can be fired before a timed reloading wait period. Jumping, when permitted, is accomplished separately, by tapping the tab key (a jet-flame animation under the tank suggests the source of the impulse). A flag is also "grabbed" when a tank not already carrying one passes over its position on the field; tanks may hold one flag at a time, and the player may drop the flag by hitting the spacebar (or other configured key). These are the main interface actions available to a BZFlag player.

The Games United League

Committed BZFlag players tend to favor specific game modes, and organized play in the form of "leagues" is common. One such league is the Games United (GU) League (www.guleague.org), which fosters play among a set of player-organized teams in capture-the-flag (CTF) style games ("matches"). The GU League was formed in 2005, and to date has recorded more than 12 thousand matches. Teamwork is cultivated, and good sportsmanship is encouraged by policies enforced by league administrators.

GU League play is strictly two-team CTF played on the HiX map (Figure 1), a square map whose dominating geographic feature is an elevated, X-shaped obstacle covering most of the map. Shots are limited to three before reloading, jumping is permitted and shots ricochet off of obstacles rather than simply stopping. Tanks move with the default movement parameters of the BZFlag game: forward velocity is limited to 25 world units, backward to 12.5. Shots travel relatively slowly compared to tank speed (though faster than the tanks themselves), so dodging shots is possible. "Superflags" (which alter game physics and thus complicate gameplay) are strictly prohibited. The only flags are two team flags used in flag capture. GU matches are played with two teams of balanced size (typically with two or three person teams), for 15, 20 or 30 minutes.[2]

In the HiX map, various square platforms and pyramidal structures support the central X, leaving four channels under each branch for tanks to drive and shoot through. Platforms at two different levels on the X may be accessed by jumping or via up/down teleporters in the four corners. The lower platforms of the X are bisected by a wall that runs most of the length of each branch. A square catwalk running the entire outside of the playing field is connected to the X at the second platform level. Team bases (Red and Purple) reside on elevated platforms at this same level supported by four tall pyramids each, and connected to the square catwalk. There are four such bases, and the entire map has 4-way radial and reflective symmetry, but only the Red and Purple bases, at opposite ends of the map's x dimension, are used in GU matches. Four additional square obstacles sit in front of each base, and four more obstacle blocks sit on top of the catwalk behind each base. A final octagonal block sits on top of the center of the X, with an octagonal pyramid over it.

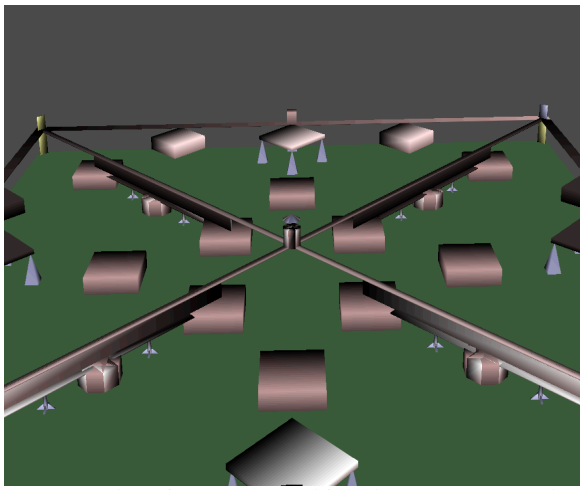

Figure 2. The HiX map, on which the GU League plays its matches.

These map features have distinct uses, and construct common pathways for movement. Since flag capture and spawning after capture involves the bases, pathways onto and off the bases are very important. These tend to be via the obstacle block in front of the base, or via the catwalk at the back. The corners of the map furthest from the opponents' base, behind the corner teleporters tend to be used for storing the flag, as it is the most protected and inaccessible spot; alternatively the obstacle block on the catwalk in back of the base is used for the same purpose. Travel to and from the base via jumping is endangered by ricochet shots off of the base-supporting pyramids, which can also be used for travel to the base, but in a way that requires considerable skill. The catwalks

to the movement system, such as strafing, but these are routinely turned down, because they would conflict with the vision for the project.

[2] All of the official games in the present corpus are 30-minute games, 15 and 20-minute games having been implemented later.

and the top of the X are often used for offensive flag-carrying, as they are relatively inaccessible, but they are also highly exposed locations when an opposing player is at the same level. The center octagonal block prevents spawn-killing after flag capture, but also presents complex ricochet possibilities to the experienced players, as do the pyramidal structures supporting the X. The center of the map is most actively used for battle, followed by the obstacle blocks in front of the bases and the four corner regions. These are just a few of the many navigational and strategic aspects of the HIX map which can be mentioned. GU players like the HIX map and the game style for its elegance and strategic subtleties.

Database and data processing

The BZFlag game server program has a built-in recording feature, which the GU uses on its servers to capture gameplay for future reference. Since 2007, more than 12,000 GU League matches have been automatically recorded by the servers; these are processed using various scripts for game details (players, observers, score) which are subsequently posted on the server owner's websites. Replay servers permit the recorded games to be viewed again, as desired. In November 2007, I obtained 2431 game files for matches which had been stored between February 4, 2007 and November 11, 2007 by one GU server administrator. These files were then processed using a specially-modified version of the BZFlag server to re-encode the information in the game files as SQL data. The data were brought into a PostgreSQL database, which was then queried using specially-written SQL functions to match the information needed to reconstruct the game's state at any given moment (e.g. by joining player names to the player position/velocity update information). These data were then processed using the R statistical programming language and environment (R Development Core Team, 2012), with additional packages for PostgreSQL connectivity (R Special Interest Group on Databases, 2009; Conway et al., 2012), 3D visualization using OpenGL (Alder and Murdoch, 2012) and network visualization (Butts, 2010).

There were four phases to the processing of this data: (1) design and population of the database, (2) development of queries and R functions to reconstruct game state, (3) checking the data for integrity, and (4) developing the statistical analysis of player movement. The first three steps are extensively discussed elsewhere (Paolillo, 2012), so we focus primarily on the analysis of movement states here.

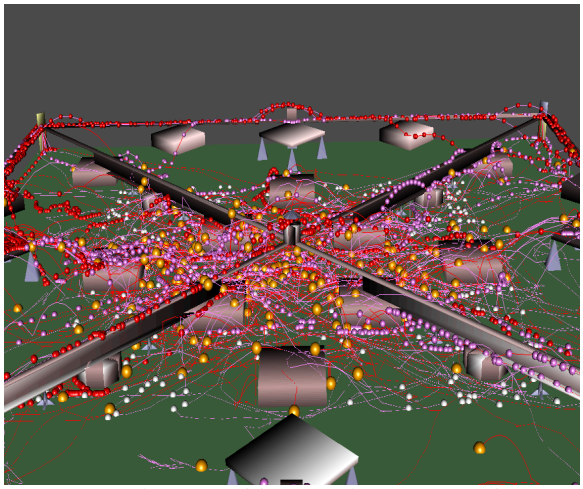

Figure 3. Player positions and movement in a GU League game. Red and purple traces represent movements of red and purple team players, respectively. Red spheres indicate locations of the red team flag, purple spheres represent locations of the purple team flag. White spheres represent player spawn points, and orange spheres represent player death points.

Analysis began by querying the activities of all player for the course of an entire game: when players connect and join a given team, when they spawn, move, fire shots, kill other players, die, and disconnect. Similar information is obtained for the team flags, representing spawns, grabs, carries, drops, and captures. Every action and event is associated with a timestamp (calibrated in milliseconds), so that the exact state of the game at any given time can be computed. Selected games were plotted in 3D visualizations as an integrity check; Figure 2 shows one such visualization, where the movements of players are represented as traces, and the locations of flags are represented by spheres. In Figure 2, the traces can be recognized as realistic in-game movement, e.g., the traces to and from the Red and Purple bases on the left and right, respectively, the circumnavigation of obstacles or via the channels, carrying team flags across the top platforms of the X, etc.

Initially, the data has no representation of game state as such: each player's movements and actions are represented independently. Moreover, the timestamps associated with different players do not necessarily match, so simultaneous relationships of different players must be established separately.[3] This was addressed by taking

[3]This is not so much of an issue for the game client, because

the original data and performing linear interpolation on the series for each player, so that positions, velocities, etc., are assigned for a matching set of time points, at 100 millisecond intervals.

The distributions of the player variables for movement were then analyzed. There are eight such variables in each player update: x, y, and z coordinates for the player in world units, an azimuth (facing) direction in radians, alongside velocities for all four, dx, dy, dz, and da. A Principal Components Analysis (PCA) of 1 million player updates showed these variables to be statistically independent — correlations among the variables are weak, and there is no more information in any given principal component than there is in one of the original variables. Consequently, PCA, Factor Analysis and related techniques cannot be used to reduce the variable space in useful ways, and an alternative means of approaching the data needs to be found.

The velocity variables are very closely associated with user actions. If an update contains a non-zero component for any of the velocities, then it can generally be assumed that the user is using the driving interface to produce movement. The only exception is when game physics intervenes, and the tank is in free-fall. On the HiX map, this only happens once the tank has jumped (or, less commonly, rolled off of a platform) and is in mid-air. The velocity variables are represented in world-centered coordinates; to understand them in terms of user actions, they need to be transformed into user-centered coordinates, via trigonometry. Examining the transformed velocity variables reveals the patterns in Figure 4.

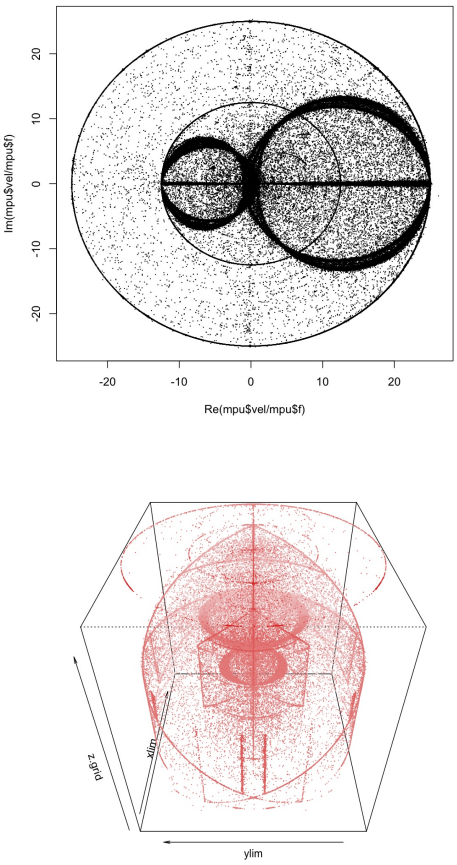

Figure 4. Velocity variables for 1 million player updates transformed into user-centered coordinates. Left: dx and dy variables; right: 3D scatterplot of dx, dy and dz.

In the top panel of Figure 4The entire plot fits in a circle of radius 25 (the maximum tank velocity) centered on the origin on the dx and dy scatterplot. Although this is maximum forward velocity the same velocity is observed in all directions, as can be seen from the outer circle of points; any such points other than straight forward (directly to the right on the x axis) represent travel in free-fall. A similar circle of points at half the radius (12.5) describes free-fall after backward jumping.

The dominant feature of the plot is a bisected horizontal figure-8 along the x axis. with the center cross of the 8 on the origin, the small lobe to the left (negative x direction, representing backward movement), and the large lobe on the right (representing forward movement). Points above the x axis represent leftward turning motion, and below represent rightward turning. Turning naturally limits forward velocity, because the forward direction changes as turning occurs. This creates the circular curve in the two lobes of the figure-8. As turning gets sharper, forward velocity falls to zero; this is because, for sharp turns, players place the mouse pointer to the left or right of the mousebox exactly on the horizontal, meaning that the tank will only turn and not move appreciably forward or backward. The bisector represents straight forward or backward movement; a relatively low density of points off of the diagonal indicates that players use intermediate velocities and turning values relatively rarely, and that full forward/backward movement and hard left or right turns are the rule.

To see the effect of free-fall on movement more clearly,

the data are sent (or recorded) as they are encountered, and the client updates the game state immediately.

the 3D scatterplot in the bottom panel of Figure 4 was created, in which the *dx* variable is rotated to be positive toward the back of the plot and negative toward the front, the *dy* variable runs left (positive) to right (negative) and the *dz* variable runs top (positive) to bottom (negative). The figure-8 is visible on the center plane, and the ring of maximum velocity values is seen in the top plane of the figure, representing updates immediately after a tank has jumped, and in helical traces around a cylinder containing the plot, starting from the top front and tracing around to the sides and back of the cylinder in both directions. These latter features represent the simultaneous changes in direction and velocity during free-fall: a tank keeps its angular momentum, and hence spins in whatever direction it happened to be turning once it has jumped, until it comes into contact with a surface again. Jump-turning is an important strategy in BZFlag for a player to quickly change direction.

Network analysis of driving states

While the velocity variables are clearly structured, to understand navigation we also need to examine their relationships in the time dimension. For this, arbitrary thresholds were established to decide if, in a given update, the tank was driving forward or backward, fast, medium or slow, whether it was turning left or right or non-turning, and whether it was vertically static or in free-fall. These thresholds and their coding is given in Table 1. Each 100 millisecond update for each player was then coded as a four-tuple of codes for these variables, the code 4-tuples being placed in sequence as they occurred in the movement in each individual player. The unspawned state ("dead") was given the special code "xxxx" to be compatible with this system.

This coding gives thirty-six sates plus one for unspawned (dead). The sequence of movement states for each game was cross-tabulated with a 100 millisecond (one place) shifted copy of the sequence, and all the games were summed together into a single 37-by-37 cross-tabulation of state-to-state transitions. This cross-tabulation is readily treated as a basis for a simple Markovian statistical model of state transitions, where each state-to-state pair is treated as an independent observation. Naturally, they are not independent, and there is a strong tendency for certain state pairs to follow others, but this nonetheless makes a good starting point for constructing a more detailed model of navigation.

We begin by exploring the nature of the state model using network mapping. For this we use the sna package for R (Butts, 2010). Because the states are thresholded versions of the velocity variables, there is considerable structure in them already that we do not want to ignore. For this reason, we avoid arbitrary graph layouts, like the Fruchterman-Reingold layout common in social network analysis, and opt for a regular grid, ordered in a manner that is easy to interpret. Since four dimensions (driving direction, turning, velocity and free-fall/surface driving) need to be represented in two, there is potential overlap in node positions and compromises had to be made. Figure 5 presents this layout, where pink states represent surface driving, green ones represent free-fall, states toward the left or right represent left or right turning, states toward the top represent forward driving, those toward the bottom represent backward, except for the 37th state "dead" (blue) which is located below the main set of states to enhance clarity (transitions to and from this state are also lightened to clarify other connections). In addition, states that differ only in velocity are placed a small diagonal distance from each other.

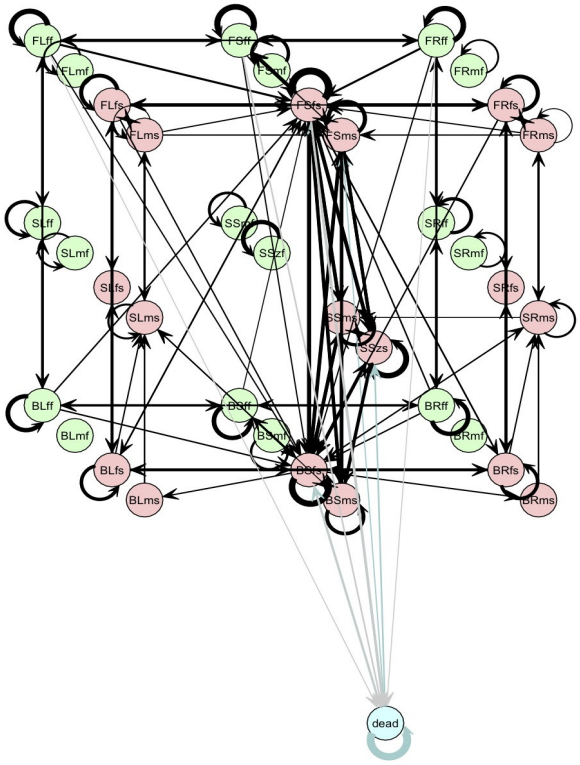

Figure 5. Network of driving states in GU League games, showing transitions with 2203 seconds (36.7 minutes) or greater of aggregate duration across 7491 total hours of driving.

Table 1
Coding for player movement variables as navigational states.

Variable	values	code	meaning
dx	$dx > 3$	F	forward
	$dx < -3$	B	backward
	$-3 < dx < 3$	S	stationary
dy	$dy > 3$	L	left
	$dy < -3$	R	right
	$-3 < dy < 3$	S	stationary
$v = (2.5 - sign(dx)/2) * \sqrt{dx^2 + dy^2}$	$v > 20$	f	full speed
	$4 < v < 20$	m	mid speed
	$v < 4$	z	zero velocity
dz	$dz! = 0$	f	free-fall
	$dz = 0$	m	surface driving

Figure 5 and 6 present the same information, except that in Figure 6, the "dead" state has been dropped, and the two systems of surface driving (left panel) and free-fall (right panel) have been plotted separately. In these figures, the surface driving states are most heavily connected to each other, centered on forward, non-turning full velocity driving (FSfs), and backward, non-turning, full velocity (BSfs); these two states have the strongest self-connections apart from the dead state, indicating that players tend to spend considerable time in them. The states most connected to these two are the stationary, non-turning states with middle and low velocity (SSms and SSzs), indicating that most driving involves no turning. Turning does take place, of course, and direct forward driving (FSfs) is connected to both left and right forward driving (FLfs and FRfs), although transitions directly between right and left driving do occur, but much less frequently. Overwhelmingly, turning states are connected to others in the same direction, though reversing direction between forward and backward driving is common (sometimes skipping the turn-only states SLfs and SRfs). Links also exist between the backward-turning states and forward non-turning driving state, and between the forward turning and backward non-turning states. These represent a common motion that players make in order to execute a turn more quickly: rocking forward and backward while turning hard to one side allows one to execute a faster turn. This can be especially useful for reversing direction (e.g. like a K-turn in automobile driving) or in aiming at an opponent who is maneuvering around one.

Mid speed driving occurs with lower overall frequency, but it has an interesting structure of its own. Self-loops exist on the mid-speed, forward driving and stationary turning states, suggesting that some time is spent in these maneuvers. The backward mid-speed driving states have no self-loops, suggesting that they are only transitional, and furthermore their transitions come only from the backward full-speed driving state and go only to stationary mid-speed turning states. There are also directional transitions to these latter states from the full-speed backward turning states, and bi-directional transitions to and from the full-speed backward driving state. This suggests a special role for the mid-speed backward states, which may be indicative of the mid-speed system as a whole. In game playing terms, full-speed driving is more convenient as it permits the player to traverse the space rapidly, and thereby accomplish game goals more quickly. However, some maneuvers are difficult to achieve while driving at full speed, and slower speeds permit more delicate coordination of driving actions.

The system of state transitions is entirely symmetrical but for a single transition from mid-speed rightward turning (SRms) to mid-speed non-turning (SSms). This suggests a possible slight right-hand bias in the system of more delicate maneuvers. We have no reason to expect that right handedness is more or less common among BZFlag players than among the general population, so a right-hand bias is not surprising in that respect. At the same time, the entire system of transitions is remarkably symmetrical with respect to right and left turning, but for this single transition, whose weight is much less than most transitions in the system. The map itself is largely symmetrical, meaning that opportunities for left and right turning situations should be similarly prevalent (as opposed to, e.g. a racing map, where laps are completed by going around a track in the same direction). This potential right-hand bias deserves to be studied fur-

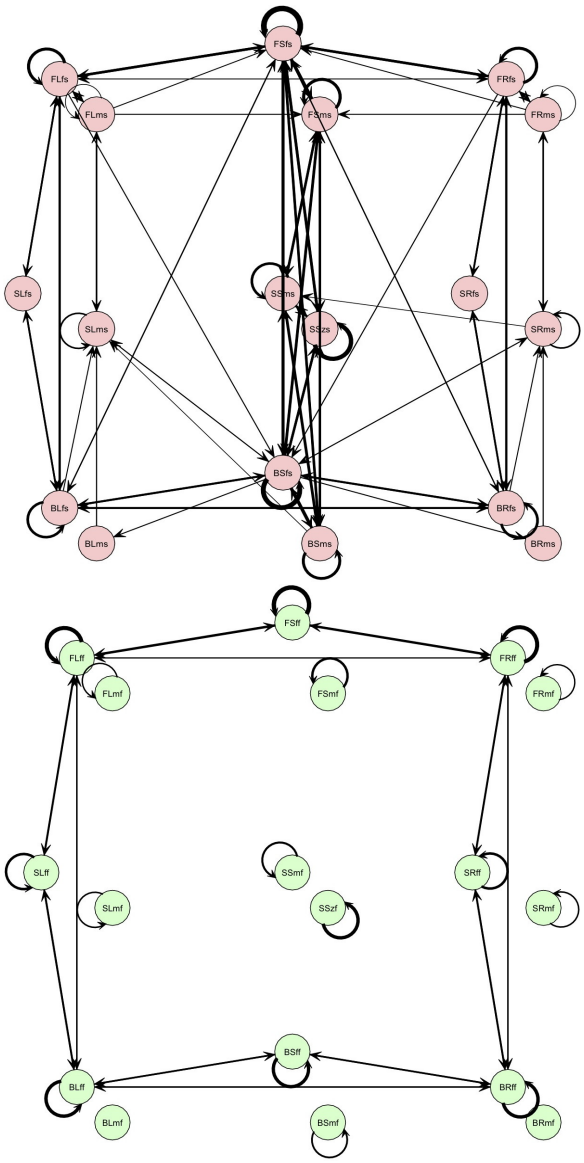

Figure 6. Network of driving states in GU League games, excluding free-fall, showing transitions with 491.5 seconds (8.19 minutes) or greater of aggregate duration across 7491 total hours of driving.

ther.

Free-fall is a much simpler system of states than surface driving, and is dominated by transitions between the full forward, forward turning, full turning, backward turning and full backward states, representing the rotation of the tank in mid-air, as governed by the game's physics. The states involving either only full turning or forward/backward motion (SLff, SRff, FSff and BSff) are typically but not always represented in this rotation The mid-velocity states are scarcely involved at all, with their strongest links to themselves, reinforcing the idea that mid-speed driving is both rarer and has different properties from full-speed driving.

Agreement of individual driving patterns

At this point, we should consider whether individual driving patterns agree with this aggregate picture. Establishing this is a large undertaking, as there are more than a thousand different league members represented in the database. Members play different games at different times of the 10-month span of the corpus, and it is possible for skill levels to have changed. Hence, these questions deserve to be investigated systematically and carefully. Spot-inspection of individual players within a game does provide some suggestion of where the answer may lie. Consider Figure 7, in which movement patterns from three players in the first game of the corpus are represented, with the same orientation as the bottom panel of Figure 4.

In Figure 7, instead of scatter-points, traces are plotted, because the updates are in natural sequence. We see the same situation as in Figure 4, but in all three panels the figure-8 is less clearly defined (if at all). As indicated in Figures 5 and 6, full-speed forward and backward movement in the surface-driving plane are strongly connected. The three panels show an interesting contrast as well, related to possible handedness bias. The top panel, representing the player labeled "Sp",[4] shows fewer traces from the top center of the figure to the right; this player apparently tends to jump while turning left. The bottom panel, representing player "TRS", shows somewhat more traces from the top center to the right, suggesting an opposite tendency to jump while turning right. The center panel, representing player "atm" shows relatively balanced left and right-turning jumps. Since these traces come from players in the course of a single game, and since small numbers of jump-turns are involved, it is possible that these observations are not fully representative. For example, it is possible that the different players had different opportunities for jumping, etc. At that same time, it is clear form Figure 7 that individual players can potentially display handedness biases, which in aggregate, could account for the asymmetric transition in Figures 5 and 6.

[4]Player names are changed here to anonymize the players.

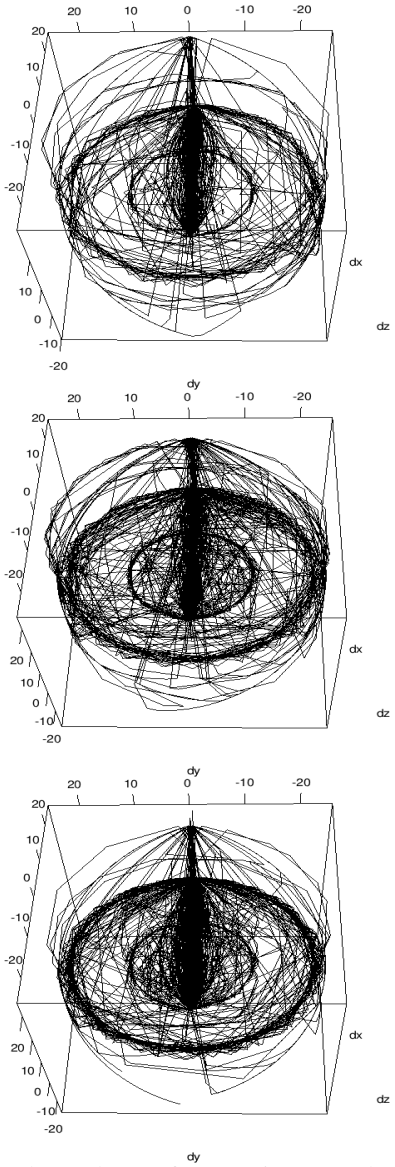

Figure 7. Three players from a given match: Sp (left), atm (middle) and TRS (right).

Discussion: Toward a statistical model

So far our investigation has been an exploration of the transitions between driving states. The observations made in this exploration can be posed as questions for a statistical analysis, as follows: (a) Are the navigational variables (direction, turning, velocity and jumping) sufficient to explain the way navigational states follow themselves, or are there specific combinations of these that are specially implicated? (b) What are the preferred navigational states on re-spawning (following death)? (c) Are there navigational states (e.g. straight surface driving, left and right turning free-fall) that preferentially lead to death? (d) Does free-fall differ from the surface-driving? How? (e) To what extent are left and right turning different, or can they be regarded as the same? (f) To what extent is the mid-speed set of states different from the full-speed set? (g) Is the apparent asymmetry in transitions from SRms to SSms significant?

These questions, and potentially others, necessitate the use of a statistical model. Different model structures could be suggested, but the one that is closest to the network visualizations used above is a log-linear model of state-to-state transition. One version of this model is given in (1).

$$\ln y_{i,j} = \alpha + \beta_i x_i + \beta_j x_j + \beta_{i,j} x_i x_j + \epsilon \qquad (1)$$

The model in (1) predicts the natural logarithm of the count y indexed by row i and column j of the cross-tabulation used in the network visualizations. Since the states are measured for regularly-spaced 100 millisecond intervals, these counts represent duration or overall time spent in a state, as well as frequency. Hence the model will tell us which states players are more likely to spend time in, and which states they spend more time going between. Using the logarithm of the count makes this a log-linear model, a member of the Generalized Linear Model (GLM) family, which is readily estimated by most statistical software and has well-understood properties (Bishop et al., 1975; McCullagh and Nelder, 1989; Agresti, 1996). The counts are estimated as a sum of an overall value α plus terms for the row i, column j and individual cell i, j. These terms are composed of a parameter β indexed for either row, column or both, multiplied by a variable or variables x representing row, column or both. A final term ϵ represents the individual error for each cell.

The α value is mainly related to the overall size of the i, j entries in the table, and is otherwise not particularly important. It is typically set to a level near the average of the logarithms of the counts i, j. Similarly, the β_i and β_j represent the overall frequency of particular states; these matter when some states are much more or less common than other states, but are otherwise not very interesting. They are also the same for both rows and columns, because the overall frequencies of states is the same, irrespective of whether we view them shifted in time. The in-

teresting term is $\beta_{i,j}x_i x_j$, which references specific combinations of row and column. In its full form for the cross-tabulation of 37 states, this term would have 1369 $\beta_{i,j}$ parameters, or as many cells as are in the model; such a term would give us a *saturated* model, with as many independent parameters to be estimated as data points to estimate them from, and would be no more economical a description than the cross-tabulation itself. Where the parameter is potentially interesting is if there are selected $\beta_{i,j}$ elements which need to be non-zero; the remaining ones can then be "pruned" from the model by setting them to zero.

Moreover, we can use other information to classify cells in the cross-tabulation, yielding a more economical model. For example, if navigational states tend to follow themselves, the loop transitions are important; these correspond to $\beta_{i,j}$ where $i = j$. In addition, each of the coded state labels represents four distinct variables, e.g., permitting us to expand the model as in (2), with respect to turning; the model can be similarly expanded with respect to the three other movement state variables.

$$\begin{aligned} \ln y_{i,j} \quad = \quad & \alpha + \beta_{i-turn}x_{i-turn} + \beta_{j-turn}x_{j-turn} + \\ & \beta_{i-turn,j-turn}x_{i-turn}x_{j-turn} \ldots + \epsilon \end{aligned} \tag{2}$$

In (2), instead of row and column terms for the specific states (combinations of four variables), we have terms that group a set of rows and/or columns together, based on the value of the turning state variable (left, right or non-turning). Note that the interaction effect for turning has nine potential combinations: LL, LS, LR, SL, SS, SR, RL, RS and RR; this makes it possible for the model to represent looping in the turning variable (by testing LL, RR and SS) independent of the other state variables. Interactions, among the four state variable can also be considered, so we can see how detailed a description of navigational state is justified.

Furthermore when estimated in a GLM, all effects, whether main or interactions effects, are estimated together so we are guaranteed that they can be independently tested for significance, using the Wald test, where "significant" means that the corresponding β is different from zero, i.e., the parameter is needed for the model to fit the data. Care must be taken to specify the variables x with a useful interpretation in mind: a staggeringly large number of variables can be created out of our original four, leading to many alternative model *parameterizations*, most of which differ by where they locate the zero *reference point* for significance tests. For the present purposes, we use dummy coding, meaning that a category variable with k categories is coded into k new variables, each new variable representing a single category, using the value 1, where the category is present. All other cells are coded with 0 for that variable. Normally only $k-1$ of these variables can be used, the absent one representing the *reference category*, for which all remaining $k-1$ variables have the value zero. Dummy coding is carried out for each value, however, and in the case of finding compact statements of certain interactions, it is sometimes helpful to move the reference category.

A log-linear model was estimated from the cross-tabulated state transition data, given in Table 2. Because of the large N (269.7 million 100 millisecond transitions), using all of our data would tend to result in significant findings for all parameters tested. This would at least partly represent spurious findings; the observations are non-independent, because they come from the same games, users, teams, etc. and they are serially correlated in time. For this reason the counts in the cross-tabulation were reduced by a factor of 10,000, corresponding to what we would obtain if we randomly sampled one in 10,000 transitions. The values in the table are large (mean cell value 19.7, max 10,049, min 0, total 26,966), but not too large in that at least some parameter values test non-significant. The model in Table 2, is one of several models run, this particular one having significance tests that allow us to answer the questions in (a) through (g). It has a high proportion of explained deviance (the residual deviance is 3528.3 on 1336 degrees of freedom; null deviance: 259866 on 1368 degrees of freedom, meaning 98.6% of the variance is explained by the 32 degrees of freedom in the model) suggesting a good fit; the model fit diagnostics, however, and in particular the quantile-quantile normal plot, show a poor fit of the residuals to a normal distribution (in particular, the tails of the residual distribution are heavier than expected). Hence, the assumptions for this type of model may be violated and we need to interpret it cautiously.

On the logarithm scale, positive parameter values (in the "Estimate" column) represent more frequent states or transitions, and negative ones represent less frequent ones. The "Std. Error", "z value" and "p value" columns represent the standard error of the estimate (its expected variability), the z value (the parameter normalized by its standard error), and the two-tailed probability of the Wald test of the z value. Chiefly the Estimate and p value columns are interpreted; the latter indicates whether

Table 2
Log-linear model of state transitions.

		Estimate	Std. Error	z value	p value
	α	-1.65863	1.03444	-1.603	0.108846
Main effects	dead	5.35324	1.13946	4.698	2.63e-06 ***
	Forward	-0.69935	0.04901	-14.270	< 2e-16 ***
	Backward	-0.84077	0.04884	-17.215	< 2e-16 ***
	Non-turning	0.17516	0.13327	1.314	0.188746
	full velocity	-4.18458	0.16292	-25.686	< 2e-16 ***
	mid velocity	-4.98555	0.12316	-40.482	< 2e-16 ***
	zero velocity	-3.73870	0.17305	-21.605	< 2e-16 ***
	free-fall	1.38113	1.01741	1.357	0.174624
loops	dead	5.52072	0.54438	10.141	< 2e-16 ***
	Forward	2.73282	0.04532	60.305	< 2e-16 ***
	Backward	1.69165	0.04890	34.596	< 2e-16 ***
	Left	3.45349	0.12108	28.522	< 2e-16 ***
	Right	3.40359	0.12125	28.071	< 2e-16 ***
	Non-turning	3.92049	0.07151	54.827	< 2e-16 ***
	Mid velocity	-2.45709	0.09479	-25.921	< 2e-16 ***
	Free-fall	4.84143	0.07074	68.441	< 2e-16 ***
	Surface driving	6.62821	1.01661	6.520	7.04e-11 ***
	From SRms to SSms	-13.00469	1275.75387	-0.010	0.991867
death (from)	Left	-1.05686	0.55523	-1.903	0.056979 .
	Right	-1.06102	0.55467	-1.913	0.055762 .
	Full velocity	2.36034	0.49208	4.797	1.61e-06 ***
	Free-fall	-2.99491	1.08474	-2.761	0.005764 **
spawning (to)	No turn	-0.28173	1.12698	-0.250	0.802600
	Left	-14.01415	347.87090	-0.040	0.967866
	Right	-14.03319	348.00766	-0.040	0.967835
	Stationary (not moving)	-1.45177	0.39907	-3.638	0.000275 ***
Other	No turn, free-fall mid/zero	0.42750	0.16551	2.583	0.009796 **
	No turn, mid velocity, free-fall	1.85632	0.12671	14.650	< 2e-16 ***
	No turn, zero velocity, free-fall	1.49491	0.15980	9.355	< 2e-16 ***
	Left, full velocity, free-fall	-0.70234	0.08615	-8.152	3.57e-16 ***
	Right, full velocity, free-fall	-0.65485	0.08575	-7.637	2.23e-14 ***
	No turn, full velocity, free-fall	2.30030	0.08840	26.022	< 2e-16 ***

a particular parameter should be interpreted, when its value is smaller than some criterion, generally 0.05.[5] In Table 2, different significance levels are flagged differently: 0.1 is indicated with a dot (.), 0.05 with a single asterisk (*), 0.01 with two asterisks (**), and anything less than 0.001 with three (***).

The parameters in Table 2 are reported in six groups: the α value, main effects, interactions representing loops, interactions involving death, transitions involving spawning (from death to alive), and other interactions. The Other category is actually much like main effects, because they only refer to the identities of the states themselves, not to transitions among the states; they represent combinations of the navigation variables that are either more or less frequent than expected by chance.

Among the main effects, we find that the "dead" state has a very high positive value, meaning that it is far more common than the other states. This is to be expected, as none of the other variables are relevant to the dead state, whereas the "alive" state is partitioned into 36 navigational states. This means that most cell counts will tend to be considerably lower than those involving "dead". Most parameter values among the other main effects are negative, meaning that their frequencies are relatively low, with the exception of turning/non-turning, which tested non-significant (left and right turning were also tested non-significant in separate tests) and free-fall, which is significantly positive, but not particularly large. All of these tendencies need to be viewed in terms of the reference category, which is surface driving and sitting

[5] Often times, smaller criterion values are used, especially Bonferroni-corrected values, when large numbers of tests are conducted, as they are here. We do not do this formally here, but note that the values for which significance is claimed are generally far smaller. Larger values approaching the criterion, such as 0.0058, are flagged as such and interpreted with caution.

still, where at least some variable value is distinct across the transition. In other words, these estimates specifically exclude loops, which are the majority of the state transitions, and are addressed by interaction parameters.

As main effects, some of the variables have significant co-occurrences which need to be mentioned (the "Other" parameter set); all of these involve free-fall, in which the physics of the game are implicated. Three parameters tell us that non-turning is more common with mid and zero speed free-fall, while the other three tell us that full velocity free-fall especially favors non-turning, with respect to both left and right turning. Of the six parameters, one (No turn, free-fall, mid/zero) is partly redundant and also somewhat near to the criterion value of significance. Hence, it appears that the significant co-occurrence of navigational variables is dictated by the game physics, and not navigational choice, although there may be a preference for non-turning jumps. Again, we need to be cautious, as much of the time spent in turning jumps is likely to be found in the loops, which are treated in interactions.

Loop transitions, i.e., those involving the same value of one or more variables, have very strong effects. However, no combinations of navigation variables were found to have significant interactions in loops, in spite of testing numerous such interactions. Hence, the navigation variables have the strongest tendency to be the same across any state transition, but there is no reason to treat any combination of navigation variables as special (e.g. hard-turn forward driving, etc.). The dead-to-dead loop transitions have a high parameter value, but this is not too surprising as there is little one can do while dead other than spawn, and the game physics sets a minimum (generally about 2 seconds) respawn time.

Looping in forward movement is favored, as is backward, but less so; here the relevant reference category is stationary (or barely moving), so this suggests that players generally chose movement over remaining still, and preferentially forward movement. Left and right turning have high and nearly identical values; they are not significantly different from each other, but aggregating left and right together in the loops would inappropriate, because left and right turning do not flip freely back and forth. Free-fall has a high value, again implicating physics, but surface driving is higher, meaning that players on the surface tend to spend more time there as opposed to jumping at every possible point.

Among the velocity variables, mid-velocity is the only significant parameter, with a moderate negative value, indicating that mid-velocity is less likely to loop than other navigation variables. In other words, mid-velocity is probably mostly transitional between full and zero velocity (or vice versa). The one apparent left-right asymmetry we noted in the network, from SRms to SSms, was tested as a separate interaction. Not only does this transition have a large negative value, it is also clearly non-significant, meaning that when other factors are controlled for, there is no evidence for any apparent handedness asymmetry in the movement data.

Transitions involving death and spawning were similarly tested. From the model, left and right turning are not significantly associated with death or spawning. Full velocity motion is (it increases a player's chances of dying), while free-fall seems to reduce one's chances of dying, contra the received wisdom of GU players that it is better to remain on the surface (and dodge shots) than it is to jump. Quite possibly, players learn when to jump better than how to dodge; the tendency to die in full forward motion also suggests a failure in dodging. From spawning, players tend to spawn in a stationary state, whereas neither turning nor any kind of motion appears to be favored. Of course, these assessments take place within a time window of one tenth of a second; it may be necessary to look at a larger time window to notice other trends.

Conclusions

The exploration of navigation states in GU BZFlag games reveals a number of tendencies, many of which are directly relatable to the physics of the game. Overwhelmingly, with a few exceptions, the navigational variables of a BZFlag player, which are directly related to interface choices, appear to be independent, except when governed by game physics. Otherwise, the primary navigational pattern we find is that players tend to remain in a navigational state for some time, irrespective of physics. We see this especially with forward and backward motion and turning. Preliminary support for some apparent tendencies related to player choice, such as handedness in left-right turn bias, appears to melt away in the statistical model. However, much more can be done in the framework illustrated by the approach taken here.

First, we have limited our attention to navigational states across a tenth of a second time window. With respect to planning and execution of tactical plans, this is a very

short period of time; any sequence of actions having the effect of advancing the goals of the game are likely to be carried out over a longer time frame, encompassing seconds, tens of seconds or even minutes. Hence we need to expand the time-scope of the model considerably. While in one sense, this could be done by considering sequences of three or more navigational states, the combinatorics of this approach makes it potentially cumbersome (e.g., for the four state variables, we could be investigating over 50 thousand combinations of transition pairs, instead of 1369 transitions). To expand up to even a full second would require a different approach. Alternatively, since it appears that the navigational variables are both independent and tend to loop within navigational states, we could focus attention on the points when these variables change value, and the durations of the states between them. This would require a somewhat different statistical approach from that used here.

Another way the model could readily be expanded relates to the potential for individual variation, as would be especially important in the handling of handedness biases and personal navigation styles. These questions are normally addressed through incorporating random effects in the model, as contrasted with the fixed effects we have examined here. Fixed effects are mostly used for variables controlled by experimental conditions, whereas random effects are needed for variables whose values arise in the process of analysis, and through which other variables may be correlated. An example of this is the individual player: different players participate in each match, some in several matches; any instance of a particular tank driving, turning left or right, jumping, etc., is related to whatever player is being observed. Each of the effects observed here could potentially be different for different players (e.g. they may have different average latencies between jumps, turns, etc.). We could partly address this concern by creating networks like that in Figure 5 or models like that of Table 2 for each individual player; random effects go beyond this by allowing us to compare across the individual models and make generalizations. Mixed-effects modeling, with both random and fixed effects, is increasingly common in social science research, and statistical packages that use mixed effects in GLMs are now readily available. Other effects that could be introduced as the model is expanded include teams and in-game events such as collisions with obstacles (or potential collisions) player death, flag capture, etc.

Returning to the questions that initiated our study, we see that, in the case of the GU games of BZFlag, players' navigation of the MOG is strongly governed by the physics and controls available to the player. While the user interface in BZFlag is designed to be simple, ostensibly to make learning it easier, we find that in being so, it heavily constrains player navigation. Nearly every pattern we can identify in the system of state transitions involves a physical constraint of the game, be it maximum velocity, free-fall or spawning latency after death. When given a choice, players appear to *always* move at the physical limits of the system, a situation that is unlike movement and navigation in the "real world" that the system ostensibly models. Moreover, given that navigation is essential to game objectives, players are effectively forced to contend with learning the controls and simulated physics system prior to being able to accomplish other goals or learning. This situation raises important questions to be answered by designers of serious games. If a 3D virtual environment is to be employed in a serious game, to what extent can it be expected to contribute to the learning goals of the game, and to what extent could it be an encumbrance? Just as flag capture can easily be described in terms of driving between the players' home and opposing bases, scavenger hunts, such as those employed in Quest Atlantis Barab et al. (2005), and other game-world based learning tasks have an inherent navigational component. More elaborate movements systems, such as coordinated mouse-keyboard systems allowing strafing and/or mid-air directional changes (as found in Quake, Second Life, etc.), should be studied using the same approach as employed for BZFlag to asses whether these systems pattern differently. At the same time, the potential learning overhead needs to be considered, as well as the relation of physics to the learning goals of the game. A trainer for operators of a specialized piece of equipment has a clear need for a game physics that closely resembles the actual circumstances in which that equipment is employed. Abstract goals, such as cooperative teamwork, or mathematical and geometric reasoning may not be so constrained, and the imposition of an arbitrary system of physical constraints is potentially a serious encumberance to the learning goals.

While the approach of the current research is not able to make strong statements regarding the role physics in its contribution to learning goals, it does provide a framework for statistically investigating the nature of navigation in a 3D game. This framework is readily adapted to other data of this nature from other games, with different physics, movement controls and game goals. Using this framework, it will ultimately be possible to answer questions such as those posed immediately above. These questions will be pursued in future studies as the methods and approach to navigation presented here are developed.

References

Agresti, A. (1996). *An Introduction to Categorical Data Analysis*. New York: John Wiley and Sons.

M.A. Ahmad, B. Keegan, J. Srivastava, D. Williams, and N. Contractor. (2009). "Mining for gold farmers: Automatic detection of deviant players in MMOGS," *Computational Science and Engineering*.

M.A. Ahmad, B. Keegan, J. Srivastava, D. Williams, and N. Contractor. (2010). "Dark gold: Statistical properties of clandestine networks in massively multiplayer online games," *IEEE Second International Conference on Social Computing (SocialCom)*, 2010.

Adler, D., and D. Murdoch. (2012). rgl: 3D visualization device system (OpenGL). R package version 0.92.880. http://CRAN.R-project.org/package=rgl

Barab, S., M. Thomas, T. Dodge, R. Carteaux, H. Tuzun. (2005) "Making learning fun: Quest Atlantis, a game without guns," *ETR& D*, 53.1, 86Ð107.

Bardzell, J., J. Nichols, T. Pace, and S. Bardzell. (2012). "Come meet me at Ulduar: Progression raiding in World of Warcraft," Accepted for *CSCW 2012*.

Bishop, Y., S.E. Fienberg and P.W. Holland. (1975). *Discrete Multivariate Analysis: Theory and Practice*. Cambridge: MIT Press.

Boellstorff, T. (2010). *Coming of Age in Second Life: An Anthropologist Explores the Virtually Human*. Princeton: Princeton University Press.

Bruckman, A.S. (1995). *MOOSE Crossing: Construction, Community, and Learning in a Networked Virtual World for Kids* (Ph.D dissertation). Cambridge: MIT Media Lab.

Butts, C.T. (2010). sna: Tools for Social Network Analysis. R package version 2.2-0. http://CRAN.R-project.org/package=sna

Castronova, E. (2005). *Synthetic Worlds: The Business and Culture of Online Games*. Chicago: University of Chicago Press.

Cherny, L. (1999). *Conversation and Community: Chat in a Virtual World*. Stanford: CSLI Publications.

Conway, J., D. Eddelbuettel, T. Nishiyama, S. K. Prayaga and N. Tiffin. (2012). RPostgreSQL: R interface to the PostgreSQL database system. R package version 0.3-2. http://CRAN.R-project.org/package=RPostgreSQL

Ducheneaut, N., and R.J. Moore. (2004). "The social side of gaming: a study of interaction patterns in a massively multiplayer online game," *CSCW'04*, November 2004.

Herring S.C., D.O. Kutz, J.C. Paolillo, and A. Zelenkauskaite. (2009). "Fast Talking, Fast Shooting: Text Chat in an Online First-Person Game," 42nd Hawaii International Conference on System Sciences.

Liang, H., R. N. De Silva, W. T. Ooi, and M. Motani. (2009). "Avatar mobility in user-created networked virtual worlds: measurements, analysis, and implications," *Multimedia Tools and Applications* 45:163Ð190.

McCullagh, P., and J.A. Nelder. (1989). *Generalized Linear Models, Second Edition*. Boca Raton: Chapman & Hall/CRC.

Paolillo, J.C., and D.O. Kutz. (2008). "Online play, online connection: a longitudinal social network analysis of BZFlag," *Meaningful Play*, October 2008.

Paolillo, J.C. (2012). "The challenge of understanding tactics in a multiplayer online game: Analyzing BZFlag games of the Games United League," *CGames 2012*, Louisville, KY, July 29-30.

R Development Core Team. (2012). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

R Special Interest Group on Databases. (2009). DBI: R Database Interface. R package version 0.2-5. http://CRAN.R-project.org/package=DBI

Taylor, T.L. (2012). *Raising the Stakes: E-Sports and the Professionalization of Computer Gaming*. Cambridge: MIT Press.