Feature-Based Projections for Effective Playtrace Analysis

Yun-En Liu, Erik Andersen, Richard Snider, Seth Cooper, and Zoran Popović
Center for Game Science
Department of Computer Science & Engineering, University of Washington
{yunliu, eland, sniderrw, scooper, zoran}@cs.washington.edu

ABSTRACT

Visual data mining is a powerful technique allowing game designers to analyze player behavior. Playtracer, a new method for visually analyzing play traces, is a generalized heatmap that applies to any game with discrete state spaces. Unfortunately, due to its low discriminative power, Playtracer's usefulness is significantly decreased for games of even medium complexity, and is unusable on games with continuous state spaces. Here we show how the use of state features can remove both of these weaknesses. These state features collapse larger state spaces without losing salient information, resulting in visualizations that are significantly easier to interpret. We evaluate our work by analyzing player data gathered from three complex games in order to understand player behavior in the presence of optional rewards, identify key moments when players figure out the solution to the puzzle, and analyze why players give up and quit. Based on our experiences with these games, we suggest general principles for designers to identify useful features of game states that lead to effective play analyses.

Categories and Subject Descriptors

K.8.0 [**Personal Computing**]: General – Games

Keywords

visual data mining, game design, games

1. INTRODUCTION

The rise of online products and services has given software creators potential access to huge amounts of behavioral data. Website logs and metrics are a good example of this. This data could conceivably be used for many tasks, such as improving an interface, testing changes, clustering or classifying users, understanding common usage patterns, and finding out which features users rely on and for what tasks.

Our primary interest is analysis of behavioral data generated from games. Behavioral data is comparatively easy to obtain as it only requires recording what players do, and does not rely on surveys, videos, or laboratory experiments. In addition, players do not know they are being observed and so behave naturally. However, without direct access to players, designers have no survey or demographic data, often making it difficult for them to understand why players behave the way they do. The sheer scale of behavioral data also makes it difficult to analyze without some form of compression; combing through ten thousand replays of player interaction with a game to answer a question is simply not possible.

To allow designers to understand behavioral data, in previous work Andersen et al. designed Playtracer, a visual datamining system for analyzing behavioral data from any game with a concept of state and state transitions. By thinking of users of a game as moving through an abstract state space and attempting to reach some particular goal states, Playtracer is able to display aggregated user behavior as a graph in order to understand common player strategies and confusions. However, Playtracer has two major weaknesses. First, it is unusable in games with large or continuous state spaces. Second, its low-level definition of states often results in visualizations where it is difficult or impossible to understand what players are thinking. Much more than this is required to improve the design of a game based on player behavior.

In this work, we address both of these problems with the addition of features to state spaces. Meaningful features serve to both compress and discretize the state space, making Playtracer usable on complex or continuous games, and allow designers to examine different aspects of their games separately or in combination. These features also allow Playtracer to analyze large amounts of behavioral data, discovering players' general problem-solving strategies, their moments of insight and learning, and their frustrations and reasons for quitting. We evaluate the effectiveness of these features by analyzing player behavior from three different games, and provide guidelines for other researchers as to what features are most likely to produce useful output.

2. RELATED WORK

Visual data mining is a powerful tool for exploratory data analysis, when the user is looking for previously unknown patterns without strong assumptions [9]. Such an approach is useful when the user is unfamiliar with the data or has vague exploration goals, making it difficult to formulate spe-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

FDG'11, June 29-July 1, Bordeaux, France.
Copyright 2011 ACM 978-1-4503-0804-5/11/06 ... $10.00.

cific experiments or analyses [12].

These visual methods are beginning to gain popularity among designers for understanding gameplay data. For example, Bungie and Microsoft made use of heat maps to determine common places of player death in Halo 3 [18, 15]. This data was used to modify the topography of the environment and strength of enemies in order to minimize unfairness and frustration. Valve used a similar approach to analyze multiplayer maps in Team Fortress 2 [1], while BioWare used heatmaps to analyze common bug locations [20]. Chittaro et al. [4] use heatmaps that track what players look at and where they spend their time in order to identify poor environment layout and player personalities. Drachen et al. use emergent self-organizing maps to analyze play data from Tomb Raider: Underworld [8]. Others have attempted to show movement through a virtual environment to analyze the flow of battle [10], identify basic player behaviors [7], and find landmarks and cluster players behavior with multidimensional scaling [17, 16].

Unfortunately, these methods are often not applicable to games without virtual environments. Researchers in other fields, however, have used visual data mining to understand user behavior in domains without natural maps on which to plot data. Lee et al., for instance, created an interactive system to visualize users' clickstream behavior in online stores [13]. Youssefi et al. use graph visualization techniques for analyzing user click data on different websites to identify popular user clickstreams and commonly visited pages [19].

In order to analyze games without virtual environments, in previous work we designed Playtracer, a generalized heatmap applicable to any game with a concept of states and state transitions [2]. Using a designer-specified distance function, Playtracer takes in lists of states that players have visited, and uses Classical Multidimensional Scaling [14] to display these states on a map along with player movement between them. Typical Playtracer output on a puzzle game without a virtual environment can be seen in Figure 1. By looking at clusters of states farther away from the goal, or areas from which players are likely to fail, designers can quickly identify likely player strategies and confusions. They can then investigate further by watching specific replays or through player interviews, if necessary.

3. FEATURES

Playtracer's primary limitation is its inability to deal with very large or continuous state spaces. Playtracing games in which players explore many states leads to unintelligibly cluttered output. One natural approach to this is to collapse similar states. Typically, this is done through clustering algorithms, such as K-means [11]. However, this leads to state graphs that are difficult to accurately interpret. If the states that belong to any given cluster are not all conceptually the same, the designer will not be able to determine what a player who visits that cluster is actually doing.

If the goal is to aggregate states which are in some sense the same, an alternative is for the designer to change the definition of a state to include only the information he wishes to examine. This can be accomplished by calculating features of each state, and then selecting one or more features

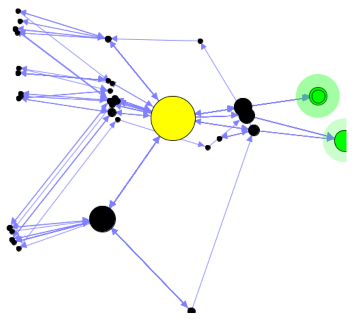


Figure 1: Example Playtracer output on a puzzle game. The yellow state is the start state, and the green states are goal states. The size of the state is proportional to the number of players who visited that state. The distance metric used tends to cluster states with similar piece placements and aligns the visualization according to how many moves it would take to reach the goal. Thus clusters of states or large states far away from the goal correspond to incorrect player hypotheses or confusions which move them farther away from solving the level.

to serve as the definition of state. Any states sharing the same features would then be collapsed together and treated as a single state. With the right selection of features, game designers can vastly simplify Playtracer's output to understand what general strategies players employ.

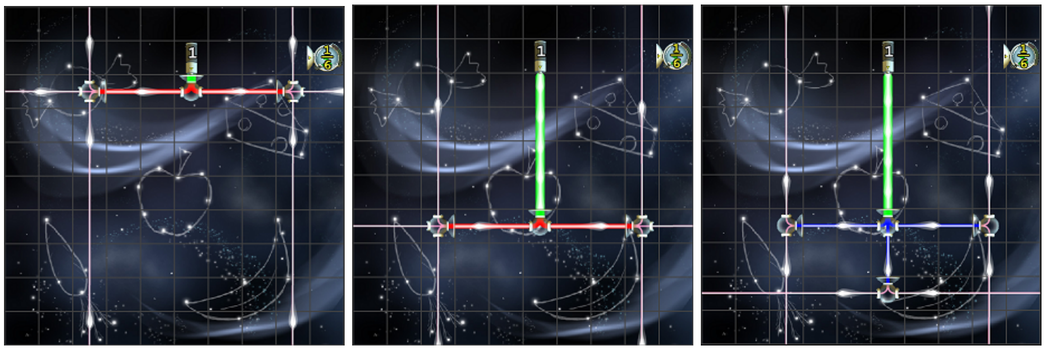
As an illustrative example, consider the educational game Refraction, shown in Figure 2. A team of graduate and undergraduate students created this game as part of a larger games for learning project. The goal of Refraction is to produce lasers with certain fractional strengths and redirect them to target spaceships. The player can accomplish this by placing pieces on the game grid to split, add, and redirect existing lasers as necessary. Each piece has fixed entrances and exits which accept lasers and output them according to certain mathematical rules; a piece with one input and two outputs, for example, takes in a laser and emits two lasers with half of the original's strength.

What features make sense in Refraction? The structure of the laser graph is one possibility, meaning that states in which pieces are in slightly different locations but producing the same lasers would be considered the same. This would capture all the mathematical operations players applied to the original laser. Alternatively, if we cared solely about the fractions players were trying to produce, and not how they were created, an even higher-level feature would be to look only at the fringe lasers. A state in which 1/6 lasers are produced by splitting first by 1/2, then by 1/3, would then be equivalent to a state in which 1/6 lasers are produced by splitting first by 1/3, then 1/2. Examples of how these features differ are shown in Figure 3.

4. UNDERSTANDING PLAYER BEHAVIOR

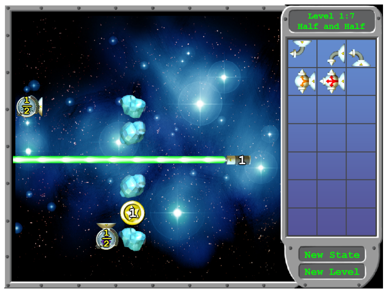
4.1 Problem-Solving Strategies

The benefit of using features instead of raw states is substantial. To illustrate this, we will use Playtracer to analyze three different facets of players behavior from different

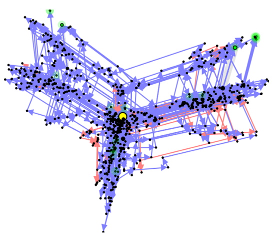


(a) (b) (c)

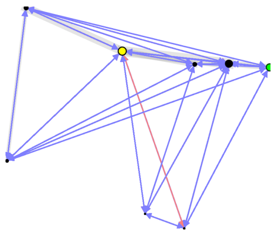
Figure 3: Examples of states and features. In the original Playtracer, the authors used piece configurations as the definition of state. Since each of these states has at least one piece in a different location, they would all be considered different. However, one might reasonably expect 3(a) and 3(b) to be equivalent given that the player is doing the same thing with a slightly different piece configuration. A feature that could accomplish this is the structure of the laser graph, with lasers being edges and pieces being nodes. States would have identical laser graphs as long as the same pieces were used in the same order on each laser. Finally, if we were only interested in the lasers players were trying to produce regardless of how they made them, we could look only at the fringe lasers shooting off into space. Since each state has six fringe lasers with strength 1/6, using the fringe lasers as our primary state feature would cause all three states to be equivalent.



(a) Refraction



(b) Old Playtracer



(c) New Playtracer

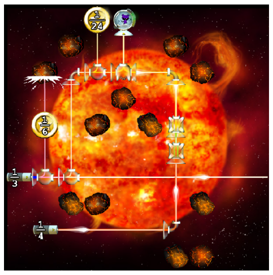
Figure 2: Refraction is an educational game where players must place pieces to split and redirect lasers to ships on the screen, shown in Figure 2(a). Defining states to be exactly the configuration of pieces on the board causes Playtracer's output to have too many states to be usable, shown in Figure 2(b). In this work, we introduce feature-based aggregation of states to analyze player behavior in complex or continuous games; Figure 2(c) is an example of how this approach can vastly simplify Playtracer's output.

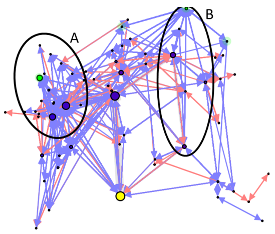
games, all of which would have been impossible without features.

First, we will use Playtracer to understand a counter-intuitive result we obtained on how players behave in the presence of optional rewards. In a previous experiment, we discovered through A/B testing that including optional coins in Refraction caused players to play significantly less during their first playthrough and return less often [3]. Players could obtain these coins by finishing the level with lasers directed through the coin; considering that players could simply choose to skip them if they so desired, that their presence would negatively affect player behavior was surprising. The experiment was conducted on a Flash game website with no access to players, making it difficult to understand exactly why they were quitting sooner.

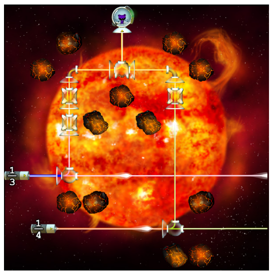
By using Playtracer to examine the behavior of players who had and did not have coins, we can begin to understand the basis for this effect. For example, the second last level of Refraction, as seen in Figure 4, has particularly tricky coins. Of the 2000 players who played the regular game, 27.4% gave up. Of the 174 players who reached the level in the no coins condition, 21.8% gave up. The solution to the level that also allows the player to win the coins is much more complex, both spatially and mathematically, than the simplest possible solution.

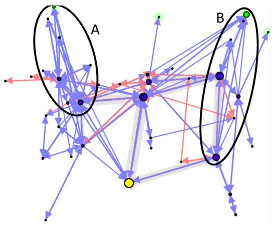
Because the coins have unique fractions, we can analyze player behavior by just knowing the fractions they are trying to produce. This is exactly the fringe laser strength feature, which represents the lasers players make at the edge of the laser graph. Thus we ran Playtracer on player data using fringe lasers as our feature, with the distance metric being the number of fringe lasers not in common between two states. The result was the visualizations in Figure 4(b). Region A corresponds to the complex solution, which entails turning the 1/3 laser into a 1/24 laser, and the 1/4 laser into


(a) Coins



(b) Coins


(c) No Coins



(d) No Coins

Figure 4: Level 53, the second last level of Refraction, and its optional coins. In our experiments on optional rewards, players with coins played less than players without coins. 4(b) shows 2000 players playing the normal game with coins. 4(d) shows 174 players from the experimental version with coins removed playing the same level. Cursory inspection shows that most players with coins restrict their exploration to complicated solutions in region A, while those without coins are willing to explore the space of simpler solutions. Thus coins can be used to strongly influence player strategies and make levels easier or harder depending on their placement.

a 6/24 laser. Region B corresponds to the simple solution, where the 1/3 laser is turned into a 4/24 laser, and the 1/4 laser is turned into a 3/24 laser.

Immediately we can see that players with coins tend to explore the region of space with a 1/6 and 1/24 fringe laser, in region A. Players in the no coins condition rarely do this; instead they spend their time assembling the easy solution. Thus, the presence of the coins is severely impacting player behavior. More specifically, they cause players to narrow their search space to solutions that satisfy them, which are more difficult and almost certainly more frustrating. This suggests that optional rewards are a very strong modulator of gameplay behavior, and can be used by designers to focus player attention on certain strategy spaces to make levels easier or harder.

4.2 Moments of Insight

While large, Refraction's state space is much smaller than the theoretically infinite state space of continuous games. This was a severe limitation of the original Playtracer, making it unable to handle games whose state spaces included any continuous variables. Using features, however, we are now able to use Playtracer on such games. We demonstrate this on two games with continuous state spaces.

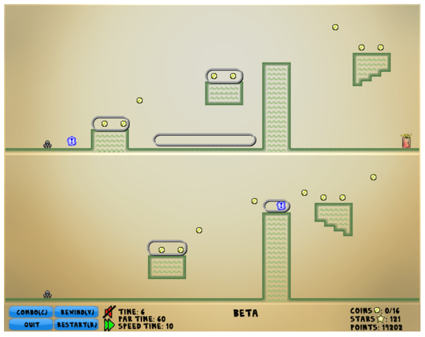


Figure 5: Hello Worlds! The character exists in multiple worlds at once. The boxes represent important progress regions, a feature we use to discretize the continuous space of character location to allow us to use Playtracer on this game. These progress regions identify key areas the player may reach that reveal important state about his progress through the level.

Hello Worlds, shown in Figure 5, is a puzzle-platformer game created by a team of undergraduate students for a video game design capstone class. The goal of the game is to guide the character through a series of spatial reasoning puzzles to collect all the stars in the game. The trick is that the character interacts with many worlds at the same time. For example, if the character hits a wall in one world, he stops in every world. The puzzle is in deciding which worlds to open and close, which influences the locations on the level the player can access.

The most obvious features of a game state in Hello Worlds are the character's location rounded to the nearest pixel, the set of open worlds, and the percentage of coins they have so far collected. If we want to analyze how players move about the environment and which world combinations they use, we could consider combining the world set and character location as our features to use for Playtracer.

Unfortunately, pixel location is too low-level to use; not only would it produce states proportional to the number of pixels on the screen(800x600), but it would produce inaccurate visualizations - two players one pixel apart are almost certainly in the same conceptual state, but would not share the same pixel location feature. Another natural approach, then, is to discretize position to a set of designer-specified key regions, or 'progress' regions within that level. Figure 5 shows such progress regions on a simple level. Four of the regions are above blocks which the character must climb to make progress. The fifth is below these blocks and shows that the character has made negative progress by falling down before they were able to reach the large wall. Players who reach the same progress region are therefore in the same conceptual state.

Thus we arrive at our features for Hello Worlds, progress region and set of open worlds. To define the distance between two states, we equally weight the distance between the two features. The distance metric for sets of open worlds is just the number of worlds not in common between the two states. The distance metric for progress regions is similar to the

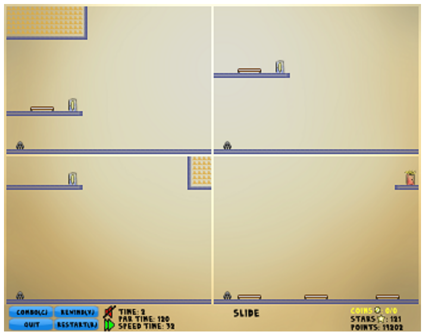


Figure 6: The level Slide in Hello Worlds. In this level, the player needs to use the blue doors to 'slide' the platforms side to side. This sliding also moves the blocks at the top of the screen. Moving the platforms and blocks into the right configuration gives the character access to the higher parts of the level, and ultimately the finish door.

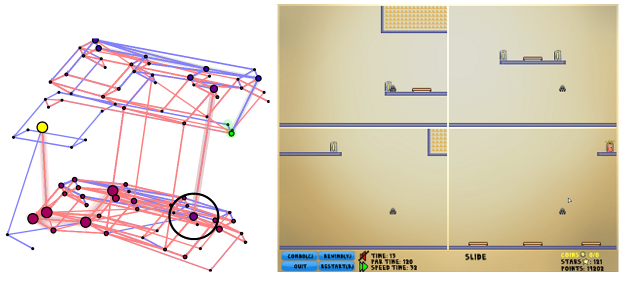
goal distance metric used in the original work on Playtracer. Each progress region is labeled by the designer with a major number representing its position on different possible paths to victory, as well as a minor number designating which path that region belongs to. We then define the distance between two progress regions as the difference in major number, or if the major number was the same, the difference in minor number. This tends to cluster together regions which players would reach in succession, as their major numbers are similar.

Combining the open worlds feature with the progress region feature allows us to generate valuable visualizations with Playtracer. Slide, shown in Figure 6, is an advanced level in Hello Worlds in which only 36% of attempts end in a successful completion of the level. To win the level the character needs to slide the platforms from left to right, and possibly back again, by opening and closing worlds with the doors on each platform. Sliding these platforms also moves the block covering the finish door. The player needs to figure out the correct configuration of slides which will both reveal the finish door and provide the necessary platforms to reach the upper levels.

Figure 7(a) shows Playtracer's output on Slide. We can see that many people are getting lost in all the sliding that needs to occur. However, there seems to be a key state that, once reached, leads to a large jump in players' chances of success to 55%. Figure 7 shows this 'aha' state in the level itself. If players reach this point in the level, it means that they have been able to successfully slide the large block which has been blocking their path to the upper tier. Playtracer's output makes it immediately visually apparent that this is the key state for players to understand how the level works. The high loss rate suggests this level should be redesigned so that players have fewer options near the start of the level, allowing them to more quickly discover this key state.

4.3 Causes of Failure

Foldit is a protein folding game designed to crowdsource protein folding [5, 6], shown in Figure 8. Discovering the structure of proteins through computational methods or through



(a) (b)

Figure 7: Playtracer output on 368 players of the level Slide from Hello World. In this visualization, states are colored based on the percentage of players who win after visiting that state. Bluer states represent a higher probability of winning, while redder states have a higher probability of losing. The circled point is the key moment of insight after which players become much more likely to win, made apparent by its bluer coloring and transition to states much closer to the goal. In this 'aha' state, players now have access to the upper platforms for the first time. Given the sudden increase in probability of winning, we can conclude that this state is the primary challenge for the players in this level.

x-ray crystallography is slow and expensive; thus, Foldit uses humans' natural spatial intuition to find the most likely candidate protein structures. In the game, players manipulate proteins to bend them into the lowest energy configuration possible, with the best solutions being forwarded to biologists to test in the lab.

Foldit's nature makes it particularly challenging to analyze. First, even small proteins can have on the order of hundreds of degrees of freedom. Second, it is difficult to define which states are equivalent, given that we are trying to compare two 3-dimensional structures which have certain constraints. Finally, there are generally no set solutions or paths to victory; players explore the space of possible protein foldings until they reach a threshold score which measures the goodness of that particular protein configuration, making it very difficult to aggregate common player strategies.

All that being said, we can still use Playtracer to gain insight into how people play introductory levels. The feature we use to compare states is the protein's contact map, a 2-dimensional array specifying which amino acids in the protein are close to each other. If element ij is present, it means amino acids i and j on the protein chain are close together. Proteins which share many entries in the contact map will be structurally similar, as the individual amino acids are in the same configuration. The most natural distance metric is simply the number of dissimilar entries in the contact map.

Figure 9 shows Playtracer's output on an introductory level of Foldit. The visualization is complex and difficult to interpret; this is not surprising, given Foldit's complexity. The visualization does convey a few key points, however. There are many states with a wide variety of structures, apparent by the many states and the wide distances between them -

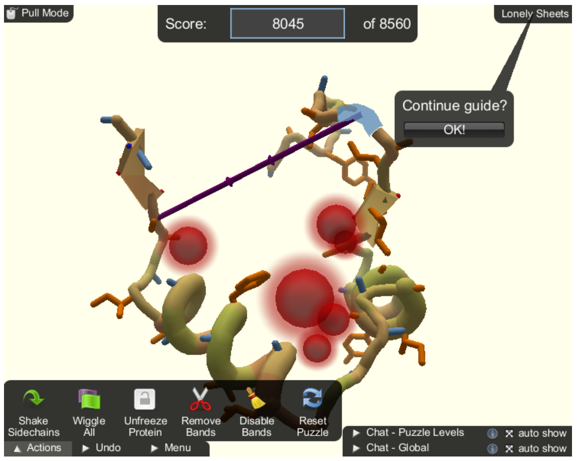
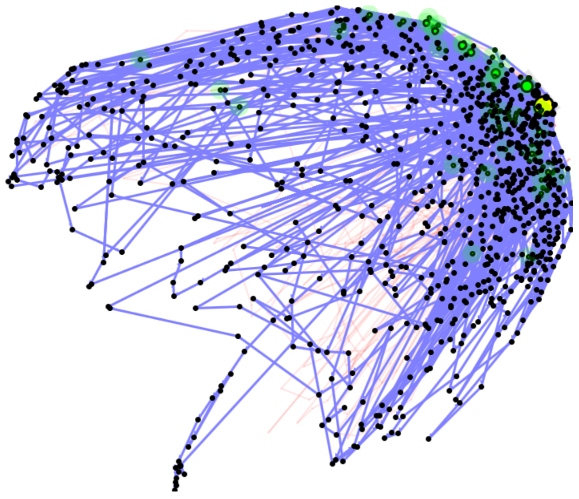


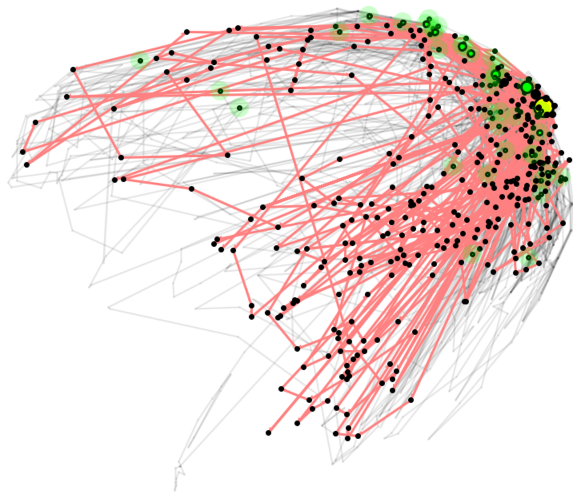
Figure 8: The scientific discovery game Foldit. Players compete to find the lowest energy structure by manipulating proteins using direct interactions and optimizations. Playtraces from Foldit are particularly difficult to analyze due to the huge state space and difficulty of comparing different structures. Even for levels in which there are set goals, designers do not know all the possible solution configurations, and every player will find a solution in a different way.

this means players tried many different ideas on this level. The pattern of movement by winners and losers also allows us to get a sense of how they do or do not differ. For example, one might hypothesize that players who quit in introductory levels did so because they could not find good protein structures. This would show up as movement away from the goal states. In reality, what we see is that both winners and losers explore regions near and far from the goal states. Losers are not simply becoming lost in the complexity of the game - something more interesting is happening. We can see that they are able to find good high-level structures based on how close they approach goal states, but can not or do not perform the last amounts of optimization to eke out the score needed to pass the level.

This visualization suggests several possible changes to this Foldit level. Given that this is an introductory level meant to teach one particular concept, the wide exploration of the search space suggests players are not following the tutorial. Some restraints on certain parts of the protein or static image guides may help constrain players to regions near the goal where they can learn the concept this level is meant to teach. In addition, because players who quit sometimes get very close to an acceptable solution without recognizing it, the game could tell players when they have found a good high-level structure and suggest they spend some time adjusting that structure instead of radically changing it. Finally, the game can remind players of the existence of its built-in optimization tools when they near a solution, preventing them from becoming frustrated with minute movements of different parts of the protein in their effort to reach the score threshold.

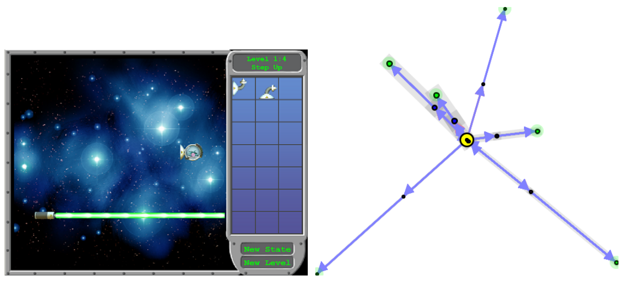


(a) Foldit winners



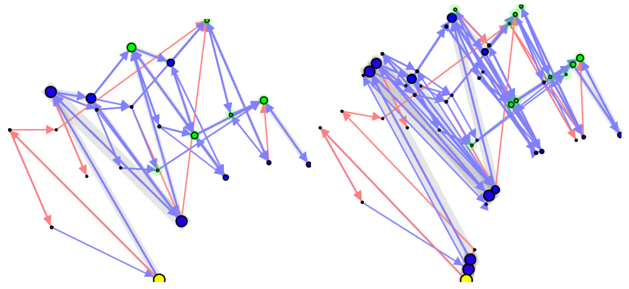
(b) Foldit losers

Figure 9: Winners and losers from 83 players of an introductory level in Foldit. These visualizations are complex, which is not surprising given Foldit's many degrees of freedom; however, we can still discern some characteristics of player behavior. Even though winning structures are near the starting configuration, players still explored wide swaths of the state space, suggesting they did not immediately see the path to victory. As such, one might guess that those who quit did so because they became lost in poor protein configurations, which would show up as movement away from goal states. However, players who eventually quit often explored quite extensively near goal structures. Thus they had the correct high-level ideas, but for some reason did not perform the remaining optimization on these 'good' structures to finish the level.



(a) (b)

Figure 10: Conceptually identical states should have the same value of the corresponding feature, or else the visualization becomes unclear. For example, using exact piece placements as our feature for Refraction results in states which represent the same strategy, yet appear in different places in the visualizations. Because each of the different spokes represent the same pattern of bending the laser up, then right, these should all be considered the same state.



(a) (b)

Figure 11: An example of the state blowup that occurs with the addition of new features. These two visualizations are generated from a level of Hello Worlds. The sole difference is that 11(b) also includes the coins feature, which represents what percentage of coins players have so far collected, discretized to the nearest 20%. This causes the number of unique states to increase, and should only be done if the designer is interested in how coins influence players' behavior in the level.

5. FEATURE SELECTION

As we have shown, extending Playtracer allows it to serve as a powerful visual datamining system for a wide array of games. Its generality and effectiveness depend crucially on the features that designers specify; incorrect choice of features will create output too cluttered to understand, or without the information the designer wishes to examine. The following three principles have proven helpful to us in our own choice of features in these games.

First, high-level features cluster states together more aggressively, making the visualizations cleaner but decreasing precision. This clustering effect is useful for data with many unique states, but may obscure what the designer wants to see. In Refraction, for example, using the fringe laser feature strips out almost all piece location information. Thus Figures 3(a) and 3(c) would be collapsed to the same state. The fringe laser metric is therefore inappropriate for analysis of players' spatial reasoning; it throws out too much information.

Second, states which are conceptually the same in some way should share the same value for the corresponding feature. Not doing this is the inverse problem of making features too general, and leads to visualizations where the same conceptual state appears in many places, unnecessarily cluttering the state space. This was a problem present in the original work on Playtracer. Defining states with exact piece placements in Refraction, for example, generates multiple paths to victory in the level shown in Figure 10, when really the paths are all equivalent.

Finally, using multiple features allows us to more accurately analyze player behavior, but multiplicatively increases the number of states. Thus designers should keep the number of features they use to the minimum necessary to understand the behavior they are interested in. Any additional features will serve only to increase the state space and obscure patterns. An example of this can be seen from Playtracer output on a Hello Worlds level, seen in Figure 11. By including the coins feature in the state definition, the number of distinct states increases several times over, making the visualization more difficult to interpret. If we are only interested in how players solve the puzzles in the level, the coins are superfluous information and should be left out.

6. FUTURE WORK

We have given guidelines as to which features are likely to be successful and their effects on the produced visualizations. However, in games with many features, it may be difficult to tell which ones to pick. One area of future work is thus automatic detection of useful features given a large set of features and some particular analysis the user wants to run. For example, if the designer wishes to understand what separates losers from winners in a level, it could be possible to generate many graphs from different combinations of features, calculate if they differ in some significant way by examining the underlying state transitions, and present only the interesting ones to the user.

Another area of future work is the inclusion of temporal and interface information. Our goal is to understand what players are thinking based on collected behavioral data. Which states players visit and in what order is a great deal of information, which can shed light on players' thoughts. However, there is information that we could leverage which we currently ignore. Long pauses indicate thought, for example, whereas quick sequences of actions may imply frustration. Likewise, frantic clicking and key-pressing would tell us the player is panicking, even though it may not alter the game state directly. We would like to include this information in Playtracer's output to better understand player intent.

Finally, while we are able to use Playtracer to analyze data gathered from several varied games, the ultimate measure of its success would be whether or not a designer unfamiliar with the tool could quickly learn it and also extract useful information from player data. Thus a study comparing the time spent and effectiveness of Playtracer to more standard data analysis or replay-watching is needed to understand how useful Playtracer is to game designers and where it fits best in the game design process.

7. CONCLUSION

In this paper we removed two key weaknesses of Playtracer, a powerful visual datamining system for games with states and state transitions, by introducing the notion of state features. These features collapse states without losing salient information and make Playtracer output significantly easier to understand. We show how this method allows us to use Playtracer to analyze previously untraceable data from the educational game Refraction, as well as two previously untraceable games with continuous state spaces, Hello Worlds and Foldit.

Designers have full control over what they view as features, and the use of different sets of features will result in output that only analyzes the facets of the game those features capture. Based on our experiences, we suggest three general principles for the selection and use of features. First, high-level features tend to collapse the state space more aggressively, which is generally desirable for complex games. Second, states should share the same value for a feature if they are conceptually the same to prevent the appearance of multiple states which are in fact identical. Finally, each additional feature causes more fragmentation of the state space, so only the minimum number of features should be used. Keeping these principles in mind when choosing features makes it much more likely that Playtracer's output will be useful for analyzing how players interact with a game.

8. ACKNOWLEDGMENTS

We thank Kongregate and the additional creators of Refraction, Hello Worlds, and Foldit: Ethan Apter, Michael Beenen, Brian Britigan, Eric Butler, Seth Cooper, Mai Dang, Happy Dong, Michael Eng, Jeff Flatten, Sean Gottschalk, Justin Irwen, Christian Lee, Marianne Lee, Emma Lynch, Stephen Sievers, and Blake Thompson. This work was supported by the University of Washington Center for Game Science, DARPA grant FA8750-11-2-0102, an NSF Graduate Fellowship, NSF grant IIS0811902, Adobe, Intel, and Microsoft.

9. REFERENCES

[1] M. Ambinder. Valve's approach to playtesting: The application of empiricism. Game Developer's Conference, Mar. 2009.

[2] E. Andersen, Y.-E. Liu, E. Apter, F. Boucher-Genesse, and Z. Popović. Gameplay analysis through state projection. In *FDG '10: Proceedings of the Fifth International Conference on the Foundations of Digital Games*, pages 1–8, New York, NY, USA, 2010. ACM.

[3] E. Andersen, Y.-E. Liu, R. Snider, R. Szeto, and Z. Popović. Placing a value on aesthetics in online casual games. In *CHI '11: Proceedings of the SIGCHI conference on Human factors in computing systems*, New York, NY, USA, 2011. ACM.

[4] L. Chittaro and L. Ieronutti. A visual tool for tracing users' behavior in virtual environments. In *AVI '04: Proceedings of the working conference on Advanced visual interfaces*, pages 40–47, New York, NY, USA, 2004. ACM.

[5] S. Cooper, F. Khatib, A. Treuille, J. Barbero, J. Lee, M. Beenen, A. Leaver-Fay, D. Baker, Z. Popović, and F. Players. Predicting protein structures with a multiplayer online game. *Nature*, 466(7307):756–760, August 2010.

[6] S. Cooper, A. Treuille, J. Barbero, A. Leaver-Fay, K. Tuite, F. Khatib, A. C. Snyder, M. Beenen, D. Salesin, D. Baker, and Z. Popović. The challenge of designing scientific discovery games. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, FDG '10, pages 40–47, New York, NY, USA, 2010. ACM.

[7] P. N. Dixit and G. M. Youngblood. Understanding playtest data through visual data mining in interactive 3d environments. In *12th International Conference on Computer Games: AI, Animation, Mobile, Interactive Multimedia and Serious Games (CGAMES)*, 2008.

[8] A. Drachen, A. Canossa, and G. N. Yannakakis. Player modeling using self-organization in Tomb Raider: Underworld. In *CIG'09: Proceedings of the 5th international conference on Computational Intelligence and Games*, pages 1–8, Piscataway, NJ, USA, 2009. IEEE Press.

[9] M. Ferreira de Oliveira and H. Levkowitz. From visual data exploration to visual data mining: a survey. *Visualization and Computer Graphics, IEEE Transactions on*, 9(3):378 – 394, 2003.

[10] N. Hoobler, G. Humphreys, and M. Agrawala. Visualizing competitive behaviors in multi-user virtual environments. In *VIS '04: Proceedings of the conference on Visualization '04*, pages 163–170, Washington, DC, USA, 2004. IEEE Computer Society.

[11] A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[12] D. A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8:1–8, January 2002.

[13] J. Lee, M. Podlaseck, E. Schonberg, and R. Hoch. Visualization and analysis of clickstream data of online stores for understanding web merchandising. *Data Mining and Knowledge Discovery*, 5:59–84, 2001. 10.1023/A:1009843912662.

[14] U. of Konstanz Algorithmics Group. MDSJ: Java library for multidimensional scaling (version 0.2). http://www.inf.uni-konstanz.de/algo/software/mdsj/, 2009.

[15] R. Romero. Successful instrumentation: Tracking attitudes and behviors to improve games. Game Developer's Conference, Feb. 2008.

[16] R. Thawonmas and K. Iizuka. Visualization of online-game players based on their action behaviors. *Int. J. Comput. Games Technol.*, 2008:5:1–5:9, January 2008.

[17] R. Thawonmas, M. Kurashige, and K.-T. Chen. Detection of landmarks for clustering of online-game players. *International Journal of Virtual Reality*, 6(3):11–16, 2007.

[18] C. Thompson. Halo 3: How microsoft labs invented a new science of play. *Wired*, 2007.

[19] A. H. Youssefi, D. J. Duke, and M. J. Zaki. Visual web mining. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, WWW Alt. '04, pages 394–395, New York, NY, USA, 2004. ACM.

[20] G. Zoeller. Development telemetry in video games projects. Game Developers Conference, 2010.