Player Modeling using Self-Organization in *Tomb Raider: Underworld*

Anders Drachen, Alessandro Canossa and Georgios N. Yannakakis

***Abstract*—We present a study focused on constructing models of players for the major commercial title Tomb Raider: Underworld (TRU). Emergent self-organizing maps are trained on high-level playing behavior data obtained from 1365 players that completed the TRU game. The unsupervised learning approach utilized reveals four types of players which are analyzed within the context of the game. The proposed approach automates, in part, the traditional user and play testing procedures followed in the game industry since it can inform game developers, in detail, if the players play the game as intended by the game design. Subsequently, player models can assist the tailoring of game mechanics in real-time for the needs of the player type identified.**

Keywords: Player modeling, unsupervised learning, emergent self-organizing maps, Tomb Raider: Underworld

I. Introduction

Being able to evaluate how people play a game is a crucial component of the user-oriented testing process in the game development industry. During the development phases, games are iteratively improved and modified towards the final gold master version, which is published. Representatives of the target audience as well as internal professional testers spend hundreds of hours testing the games and evaluating the quality of the gaming experience [1]. Moreover, one of the key components of user-oriented testing both during production, as well as after game launch, is to evaluate if people *play the game as intended* — and if not, to find out why there is a difference between the intended and actual playing behavior, and whether this has an impact on their playing experience [1], [2]. Given that nonlinear game design (i.e. game design in which the player has multiple choices about how to progress in the game) becomes increasingly popular — massively multi-layer on-line games being a good example of the increased popularity of nonlinear *sandbox*-type games — the need of more reliable and detailed user-testing is growing.

Within the last five years, instrumentation data — or *game metrics* as they are referred to in game development — has gained increasing attention in the game industry as a source of detailed information about player behavior in computer games [2]. Gameplay metrics are detailed numerical data extracted from the interaction of the player with the game using specialized monitoring software [3]. The application of machine learning on such data and the inference of playing patterns from the data can provide an alternative quantitative approach to and supplement traditional qualitative approaches of user and playability testing [4].

In this paper, we investigate dissimilar patterns of playing behavior in the popular commercial game title *Tomb Raider: Underworld*[1] (TRU) using the principles of self-organization. The experiments presented are based on a data set derived from 1365 players, which completed the entire TRU game during November 2008. Data was collected via the EIDOS Metrics Suite (a game metrics logging system utilized by EIDOS). The data collection process is completely unobtrusive since data was gathered directly from the game engines of subjects playing TRU in their natural habitat (via the Xbox Live![2] web service) rather than in a laboratory setup.

Six statistical features that correspond to high-level playing behaviors are extracted from the data. Information about game completion time, number of deaths, causes of death and help on demand actions is used for feature extraction. An initial analysis via k-means and Ward's hierarchical clustering methods provides some first insight into the inner structure of the features. Then unsupervised learning via Emergent Self-Organizing Maps (ESOMs) [5] — an efficient visualization tool for large-scale self-organizing maps (SOMs) [6] — is used to identify dissimilar clusters (types) of playing behavior. The highest-performing ESOM network which is built on the six individual features of play projects four clusters of behavior, covering the vast majority of the sampled players. The four groups of behavior identified by self-organization are subsequently labeled in terms of the game mechanics and game design. This process ensures that the outcome of the analysis is in a form and terminology that is usable by the game designers evaluating if the players interact with the game in the way intended (given the specific input playing features) and deciding on whether and how to change the design in order to alter the behavior of the players.

Unsupervised learning has been utilized for modeling player behavior in games (e.g. in [7]); however, to the best of the authors' knowledge, SOMs have not been applied for modeling high-level behaviors of players trained on data of completed games. Results showcase the effectiveness of unsupervised learning in capturing dissimilar high-level playing behaviors (e.g. completion time) in the TRU game. Such behavior clusters can form the basis of further investigation in lower-levels of playing behavior (e.g. shooting accuracy in a specific level of the game).

AD and GNY are with the Center for Computer Games Research, IT University of Copenhagen, Rued Langgaards Vej 7, DK-2300 Copenhagen S, Denmark (email: {drachen, yannakakis}@itu.dk). AC is with IO Interactive, Kalvebod Brygge 35-37, 1560 Copenhagen V, Denmark and with the Denmark School of Design, Strandboulevarden 47, Copenhagen, Denmark (email: alessandroc@ioi.dk; aca@dkds.dk).

[1] http://www.tombraider.com

[2] http://www.xboxlive.com

Because this study is based on *large-scale* data collection (1365 players) obtained from a major commercial game and via an industrial logging system, the findings of this study directly address the requirements for player behavior modeling in the game industry. Very few studies of academic game artificial intelligence (AI) investigate actual commercial-standard games due to lack of source-code accessibility and even fewer examine the game as a whole. This often generates challenges with respect to the generalization of the main findings to a real game production and with respect to directly addressing the needs of the game industry.

II. Player Modeling

User modeling is a broad field of research with numerous applications. In this section we will concentrate on user modeling approaches to games, namely player modeling [8], [9]. Among the few player modeling studies existent in the literature, quantitative models of players have been built to assist the learning of basic non-player character (NPC) behaviors (e.g. moving, shooting) in Quake II [10], [7], [11]. In those studies self-organizing maps [7], bayesian networks [11] and neural gas [10] approaches are utilized for clustering game-playing samples. Similarly, self-organizing maps have been used for clustering the trails (player waypoints) of users playing a simple level exploration game [12]. Viewing player modeling as an intermediate process of adaptive learning in games, Yannakakis and Maragoudakis [13] trained naive bayesian models of Pac-Man players which infer values for parameters of an on-line neuro-evolutionary learning mechanism. The on-line learning mechanism was designed to maximize the entertainment value of the game by adjusting NPC behavior during play. Within the field of interactive narrative and AI in games, quantitative models of players — partially built on theoretical qualitative models — have been used to dynamically select the content of an interactive story [14].

Self-organizing maps — as a tool for player modeling — have been primarily used to generate low-dimensional data maps to assist training of NPC behaviors. For instance, the trained SOM can generate the desired outputs of a supervised learning approach [7] or form the action-state space of a reinforcement learning approach [15]. On that basis, Thurau et al. [7] utilize SOMs to lower the dimensionality of input data for training multi layered perceptrons that perform simple playing behaviors in Quake II while White and Brogan [15] use SOMs to generate the state space of a temporal difference learning mechanism which learns to play RoboCup simulated soccer.

All the aforementioned studies focus on constructing models of playing behavior based on small-scale player-data collection experiments held in laboratories. Also, the vast majority of approaches concentrate on a few specific scenarios (e.g. imitate human movement in a particular level of a game) while the game environments investigated are in-house instrumented test-bed games or simplified versions of commercial games. This results to the simplification of the learning task which acts in favor of the learning approach; however, the scalability of the obtained performance is often questionable.

Fig. 1. Screenshot from Tomb Raider: Underworld.

This paper differentiates in that high-level behaviors of players that completed a game are modeled; data is gathered in a natural setup via an industrial logging system and a commercial web service; the data collection experiment consists of over 1300 players making it large-scale and representative; and the test-bed game used is a published game from one of the major franchises in the industry. Deriving models of playing behavior under these conditions appears to minimize any limitations of scalability and commercial-game practicality and contributes toward bridging the existing gap between academic and industrial game AI.

III. Tomb Raider: Underworld

The popularity of the Tomb Raider series is mainly due to the game protagonist, whom the player controls: Lara Croft. She is a combination between an action heroine and Indiana Jones, who travels to exotic locations and enters forgotten tombs and lairs, solving puzzles and finding ancient treasures at the same time. The Tomb Raider game environments have been 3D from the beginning, and *Tomb Raider: Underworld* (TRU) is no exception. The game features exceptional graphics and takes full advantage of the graphics processors of game consoles.

The game is played in third-person perspective with a flexible camera system. TRU is in essence an advanced platform game, where the player has to apply strategic thinking in planning the 3D-movements of Lara Croft, in order to solve a series of puzzles and navigate through complicated levels (see Fig. 1). Apart from the continuous risk of falling from the heightened platforms that Lara Croft needs to jump at (and from) and navigate through, the player regularly encounters different enemy types, notably animals, various kinds of monsters and mercenaries. An additional threat to the player is the environment itself. Falling into a trap, catching fire or drowning into the water are the three of the typical dangers the player is facing which are caused by the game environment.

IV. Data collection

The gameplay data, namely *game metrics*, utilized in this study were recorded using the EIDOS Metrics Suite software embedded to the TRU game. The suite is an instrumentation system which is designed to record game metrics from EIDOS games in production and post-launch, transmitting the logged data to an SQL-server via an ETL process. Game metrics are normally logged as sequences of events — with multiple types of data captured for each event — each carrying its own time stamp as well as any other pertinent contextual information. From the server, data can be extracted for analysis and visualization, creating reports for the interested parties within the game development process (e.g. quality assurance, game design, production and marketing departments).

Data from the TRU game were extracted from the SQL server system and preprocessed. The dataset used for the experiments reported in this paper contains *live data*, i.e. data from people playing the finished, published version of TRU in their natural habitats. This unobtrusive way of user data collection provides data free from bias induced by using a laboratory setup, i.e. avoids modifying the habitat of the participant [16]. The dataset used was collected during November 2008, and includes entries from 25240 players. Note that at the moment of writing there are over 1 million recorded gameplay sessions of TRU which will form the basis for future research. The 1365 of those 25240 players that completed the game – i.e. played through all levels of the game — were isolated and used in the experiments presented in this paper. TRU consists of seven main levels plus a prologue. Each game level is sub-divided into *map units*, of which there are 100. The EIDOS Metrics Suite stores data for all these map units and levels individually; however data is aggregated at the level of a completed game in this study.

A variety of different playing characteristics (game metrics) are collected for internal analysis work carried out by Crystal Dynamics[3]. Several gameplay features are logged for each player, such as the completion time of each game level, the number of times the player died, as well as the 3D coordinates of the player. For the current study, initially the focus is on defining the primary playing features — among those collected — that relate to the core mechanics of the game and, furthermore, may have an impact on the playing behavior. For instance, the ability of players to perform jump actions without dying is of key interest for classifying different playing styles, as jumping is a key TRU game mechanic.

A. Extracted Features

Six, in total, gameplay features are extracted from the data collected and are further investigated in this paper. All features are calculated on the basis of completed TRU games in this initial study. The selection of these particular features is based on the core game design of the TRU game and their potential impact to the process of distinguishing among dissimilar patterns of play.

- **Causes of death:** TRU features a variety of ways in which players can die, which can be grouped into three overall categories that encompass all possible ways players can die. The total number of times a player died for each of the three following categories and the corresponding percentages over the total number of deaths are calculated:
  - Opponent; the percent of total number of deaths caused by any computer-controlled opponent existent in the game over the total number of deaths, D_o. Dying from opponents comprises 28.9% of the total number of deaths across the 1365 data samples. The best player died only 6.32% of the times from opponents, while 60.86% is the maximum value observed for D_o.
  - Environment; the percent of total number of deaths caused by the environment over the total number of deaths, D_e. Environment-related causes of death include player drowning, being consumed by fire, or killed in a trap, comprising 13.7% of the total number of deaths across all players. The best player died 2.43% of her total number of deaths from environment-related effects, while the highest recorded value of D_e is 45.31%.
  - Falling; the percent of total number of deaths caused by a failed jump over the total number of deaths, D_f. Dying from falling comprises 57.2% of all death events making it the dominating cause of death in TRU. This is expected since the core of the gameplay consists of jumping, climbing and navigating in 3D environments. The minimum and maximum values of D_f are 27.19% and 83.33% respectively.
- **Total number of deaths:** The total number of deaths of each player, D. A total of 190936 deaths were recorded, giving a mean value of approximately 140 per player, with the best and worst player dying 16 and 458 times, respectively.
- **Completion time:** The time (in minutes) required for each player to compete the game, C. A total of 521.6 days of playtime were recorded. The average completion time of the game in the sampled data is 550.8 minutes, with specific levels generally taking longer to complete than others as they are bigger in extent and/or contain harder-to-solve puzzles. The C value varies from 2 hours and 51 minutes to 28 hours and 58 minutes, showing considerable variance which, in part, showcases the variation in the experience level, skill and play-style of the players.
- **Help-on-Demand:** The number of times help was requested, H. A key feature of TRU is the focus on puzzle solving. A typical puzzle could be a door which requires specific switches to be pressed in order to open. There are more than 200 registered puzzles in the game, which the players have to solve in order to progress through the game narrative (see Fig. 2). The game

[3] Crystal Dynamics is the developer of Tomb Raider: Underworld; http://www.crystald.com/

Fig. 2. Example of a puzzle in Tomb Raider: Underworld. Several gear wheels fit together in a specific way and the player must figure out how to manipulate them.

features a native Help-on-Demand (HoD) system which players can consult in order to get help with solving the puzzles. The player can either request a hint about how to solve the puzzle or a straight answer. The H value incorporates the total number of times that each player requested either a hint or an answer. The data from the 1365 players reveal that players generally either request both hints and answers from the HoD-system, or no help at all, for specific puzzles. It was therefore decided to combine the hint and answer requests into the H aggregated value. The H value ranges from 0 to 148, with an average of 29.4 per player.

V. Emergent Self-organizing Maps

The self-organizing map (SOM) [6] or (Kohonen map) iteratively adjusts a low dimensional projection of the input space via vector quantization [17]. A SOM consists of neurons organized in a low (2 or 3) dimensional grid. Each neuron in the grid (map) is connected to the input vector through a d-dimensional connection weight vector $\mathbf{m} = \{m_1, \ldots, m_d\}$ where d is the size of the input vector, $\mathbf{x}$. The connection weight vector is also named *prototype* or *codebook* vector. In addition to the input vector, the neurons are connected to neighbor neurons of the map through neighborhood interconnections which generate the structure of the map: rectangular and hexagonal lattices organized in 2-dimensional sheet or 3-dimensional toroid shapes are some of the most popular topologies used.

SOM training can be viewed as a vector quantization algorithm which resembles k-means [17]; however, what differentiates SOM is the update of the topological neighbors of the best-matching neuron: the whole neuron neighborhood is stretched towards the presented input vector. The outcome of SOM training is that neighboring neurons have similar weight vectors which can be used for projecting the input data to the two dimensional space. SOMs can be used for clustering of data (unsupervised learning) for the aforementioned properties. For a more detailed description of SOMs, the reader is referred to [6].

The power of self-organization — which generates emergence of structure in the data — is disused when small SOMs are utilized. The topology preservation of the SOM projection is of little use and the advantage of neighbor-neuron relation is neglected which makes a small SOM almost identical to a k-means clustering algorithm. Using large SOMs — called Emergent Self-Organizing Maps (ESOM) [5] to emphasize the distinction — and reliable visualization techniques help in identifying clusters in the low-dimensional projection of the data.

The topology size of an artificial neural network (ANN) is related to performance. A too small ANN may generate low approximation of the underlined function while a too large ANN may overfit the data when using supervised learning (e.g. multi layered ANN backpropagation training). Unlike supervised learning, ESOM size does not affect the model's performance in the same way because the neurons are restricted by the topology preservation of the map. The use of more neurons (larger maps) results to an increase of the map resolution deriving from the projection of the input space into 2 dimensions. However, there is a balance between resolution and computational effort that the ESOM designer should keep.

We use the batch algorithm for training the ESOM[4]. Batch training searches the map for finding the neuron (namely, the best-match) with a corresponding connection weight vector that matches each input vector. A best-match neuron is a neuron for which there exists at least one input vector for which the Euclidean distance to the weight vector of this neuron is minimal. In ESOM batch training, similarly to batch backpropagation, all input samples are presented to the network before the weight update if performed. The toroid topology is used to avoid border effects that are generated by clusters existent in the border of 2D sheet maps. Neurons are interconnected within the map in a rectangular grid (i.e. each neuron has four immediate neighbors). The hexagonal grid (i.e. six immediate neighbor neurons per neuron) is not preferred since recent studies indicate that the shape of the map has a greater impact to SOM performance than the number of immediate neighbors [18].

There are numerous measures proposed in the literature used to evaluate the performance of a clustering approach. Even though all measures provide dissimilar indications for the properties of the generated clusters, no measure can guarantee approximation of the performance with high accuracy. In this study we choose the average quantization error and the topographic error [6] as measures of ESOM training performance. In the case of ESOM, the average quantization error equals $\frac{1}{N}||\mathbf{x} - \mathbf{m_c}||$ across all N data samples, where $\mathbf{m_c}$ is the weight vector of the best-matching neuron. Topographic error measures topology preservation of the map and is calculated as the proportion of all input data vectors for which the first and second best-matching neurons are not adjacent [6].

[4] The databionic ESOM software tool [19] is used for training and visualizing the ESOM.

VI. Results

This section presents the main findings of the clustering approaches applied to the data. A pre-processing analysis of the data is complementary to and followed by the design of the ESOM approach for unsupervised learning of the data and the identification of the different player styles.

A. Pre-processing and Initial Cluster Analysis

All six features extracted are uniformly normalized into $[0, 1]$ before any clustering analysis is followed. Note that the *cause of death* features (D_o, D_e and D_f) are already normalized in $[0, 1]$ being percentages of the total number of deaths.

To get some first insight of the possible number of data clusters existent in the data, we apply the k-means clustering algorithm to the normalized data for all k values less than or equal to 20. The number of player observations (6-dimensional vector samples) and the sum of the Euclidean distances between each player instance and its corresponding cluster centroid (quantization error) are calculated for all 20 trials of the k-means algorithm. The analysis shows that the percent decrease of the mean quantization error due to the increase of k is notably high when $k = 3$ and $k = 4$. For $k = 3$ and $k = 4$ this value equals 19.06% and 13.11% respectively while it lies between 7% and 2% for $k > 4$. Thus, the k-means clustering analysis provides the first indication of the existence of 3 or 4 main clusters within the data.

An alternative approach to k-means for cluster analysis is through hierarchical clustering. This approach seeks to build a hierarchy of clusters existent in the data. The squared Euclidian distance is used as a measure of dissimilarity between data vector pairs and Ward's clustering method [20] is utilized to specify the clusters in the data; the resulting dendrogram is depicted in Fig 3. (A dendrogram is a tree-like diagram that illustrates the merging of data sets into clusters. It consists of many U-shaped lines connecting the clusters while the height of each U represents the squared Euclidian distance between the two clusters being connected.) Depending on where the designer sets the squared Euclidian distance threshold, T, a dissimilar number of clusters can be observed. For instance, 3, 4 and 5 clusters of data can be identified if $6.56 > T > 4.72$, $4.72 > T > 4.25$ and $4.25 > T > 3.74$, respectively.

Both clustering approaches demonstrate that the 1365 players' feature vector can be clustered in a low number of different player types. k-means statistics provide indications for 3 or 4 clusters while the Ward's dendrogram shows the existence of 2 populated and 2 smaller clusters, respectively, in the middle and at the edges of the illustration resulting to four clusters. By further splitting the populated middle-left cluster (Fig. 3), a number of six clusters can be obtained within a small difference of the squared Euclidian distance: the difference between the minimum T for obtaining 4 clusters and the maximum T for obtaining 6 clusters equals 0.52. This initial cluster analysis provides the first insights into the spatial structure of the data, which reveals the existence of a small number of clusters in the dataset; however, the analysis does not reveal anything about whether the players included in the clusters match those obtained via the ESOM (as presented in Section VI-B). The procedure followed through k-means and Ward's clustering is viewed as necessary data pre-processing prior to learning from data via self-organization.

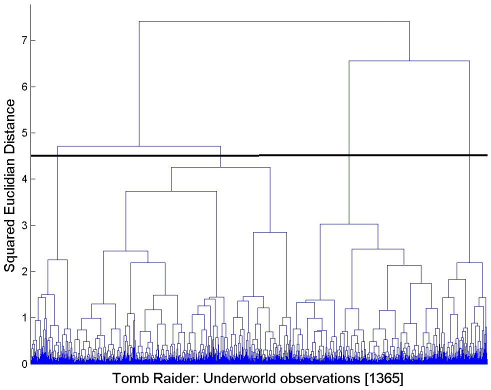


Fig. 3. Clustering of data using the Ward dendrogram method. A T value of 4.5 (illustrated with a horizontal black line) reveals 4 clusters.

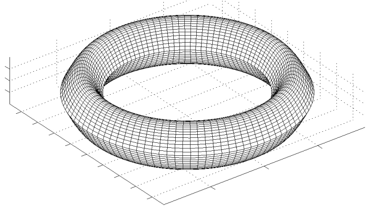


Fig. 4. Rectangular lattice map organized in a toroid shape; SOM size: 50×100 neurons.

B. ESOM

As previously mentioned in Section V, a toroid-shaped ESOM structured in a sheet grid is utilized to identify classes of playing behavior in TRU. Fig. 4 depicts the 50×100 neuron structure used in the experiments presented in this paper. The number of 5000 neurons is chosen to provide a good compromise between training performance, computational effort and resolution detail of the map. The toroid shape of the map is selected due to border effects observed in training attempts with 2-dimensional sheet maps.

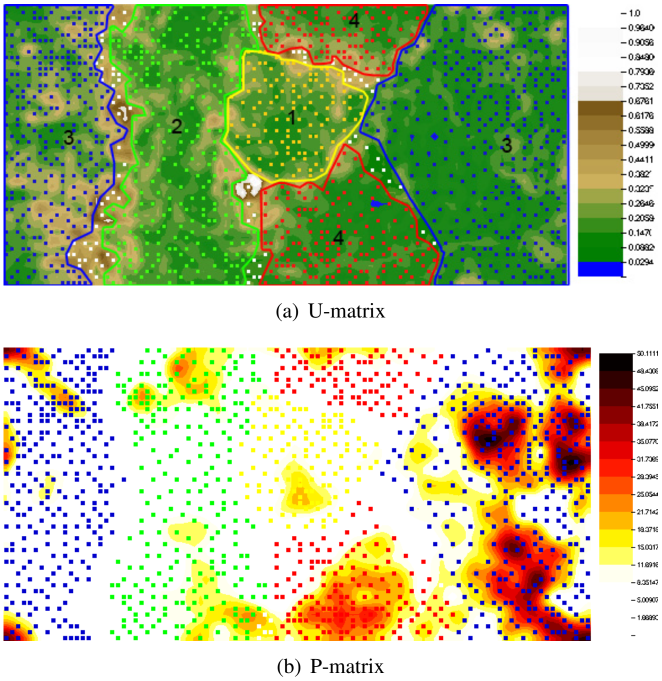
1) ESOM training: The weight vector of the ESOM is randomly initialized using a Gaussian distribution; the mean and standard deviation of the distribution are set to the respective values of the corresponding input feature. The initial neighborhood size is set to 25 which linearly drops to 1 while the weighting neighbor function, h, used is the Gaussian kernel. The learning rate is set to 0.7 but is decreased linearly during training reaching the value of 0.1 at the end of the 100 training epochs used. The training samples are presented in a randomly permuted order at each epoch of the algorithm.

In order to minimize the effect of non-deterministic selection of initial weight values we repeat the training 20 times — using dissimilar initial weight vectors — and select the ESOM with the smallest average quantization error (see Section V). The highest-performing ESOM that is examined in the remained of this paper has a quantization error of 0.038 and a corresponding topographic error of 0.005. As a baseline performance to compare against, the mean quantization and topographic error for 10 randomly generated ESOMs equals 0.1744 and 0.9983 respectively.

2) ESOM visualization: The training data can be clustered by observation of the best performing ESOM. The U-matrix depicted in Fig. 5(a) is a visualization of the local distance structure in the data placed onto the two-dimensional map. The average distance value between each neuron's weight vector and the weight vectors of its immediate neighbors corresponds to the height of that neuron in the U-matrix (positioned at the map coordinates of the neuron). Thus, U-matrix values are large in areas where no or few data points reside, creating mountain ranges for cluster boundaries. On the other hand, visualized valleys indicate clusters of data since small U-matrix values are observed in areas where the data space distances of neurons are small.

Distance based map visualizations (e.g. U-matrix) usually work well for clearly separated clusters; however, problems may occur with overlapping clusters. Density-based SOM visualizations display the density of the data onto the map space via the best-matching neurons. The P-matrix (see Fig. 5(b)) displays the local density measures with Pareto Density Estimation [21]. Neurons with large P-matrix values are located in dense regions of the input vector space and, therefore, areas with height P-matrix values indicate clusters in the data.

3) Player Types: The two map visualizations are complementary and used for cluster identification within the TRU data. Four main classes (player types) can be identified as depicted in Fig. 5(a) and Fig. 5(b). The best-matching neurons for all 1365 input vector samples are also represented with small squares of varying color — different colors correspond to different clusters of the best matching neurons. On the same basis, Table I presents the number of observations (i.e. players completed TRU) and percent of neurons belonging to each of the four clusters. Note that 87 (6.37% of the sample) players were not assigned to any cluster since their best-matching neurons are placed in cluster borders of the ESOM (see Fig. 5).



(a) U-matrix

(b) P-matrix

Fig. 5. Visualization maps of the highest-performing ESOM obtained and the 4 clusters identified. The 1365 best-matching neurons are drawn as squares on top of the maps. The maps illustrated are border-less since the map grid is organized in a toroid shape.

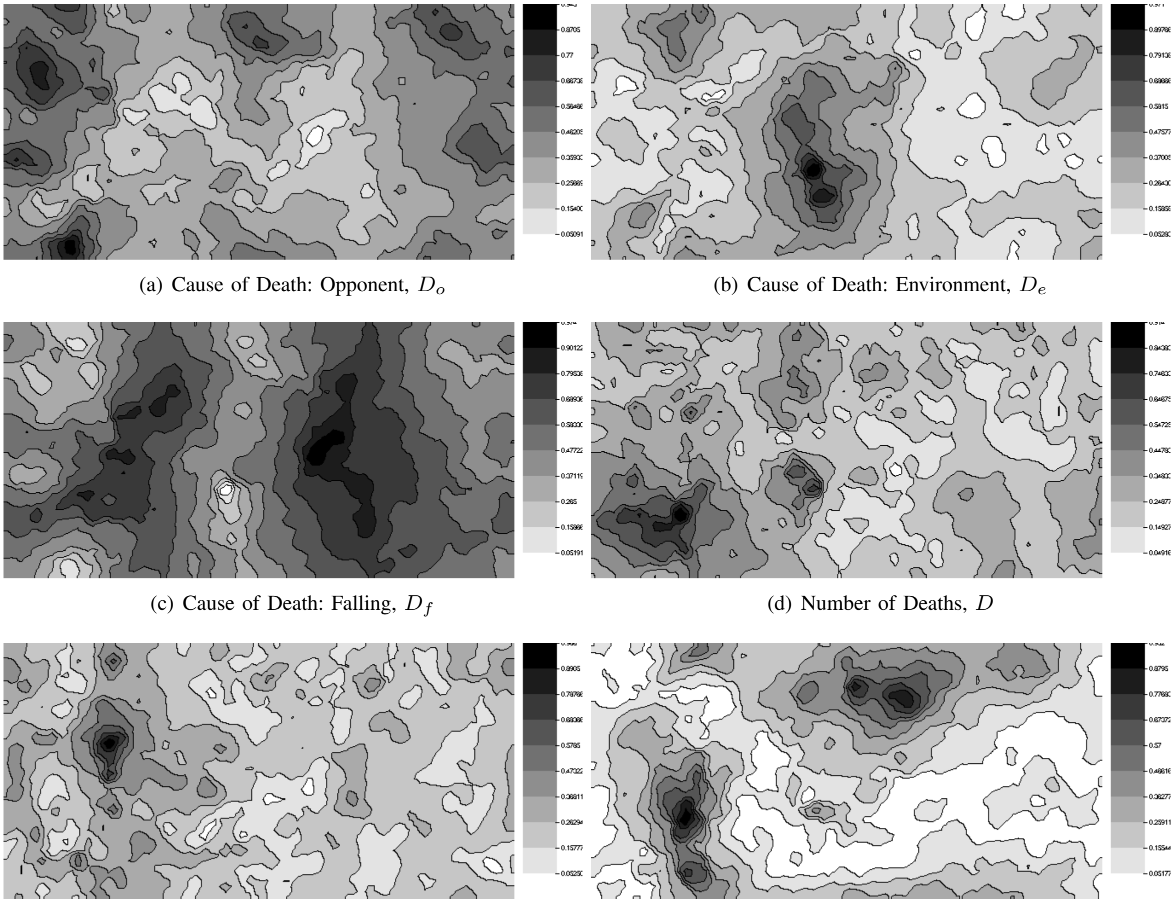
TABLE I
THE FOUR PLAYING BEHAVIOR CLUSTERS IDENTIFIED USING ESOM

Class	Observations	Neurons on map space (%)
1	122	8.68
2	270	22.12
3	641	46.18
4	245	16.56
N/A	87	6.46

Fig. 6 illustrates the corresponding component planes of the ESOM. A component plane projects the relative distribution of one input data vector component (i.e. input vector dimension) to the ESOM. In the grayscale illustration of those values, white areas represent relatively small values while dark areas represent relatively large values. By matching the component planes with the U-matrix of Fig. 5(a) we can infer characteristics (i.e. playing behavior features) for each cluster identified.

Cluster number 1 corresponds to players that die very few times; their death is caused mainly by the environment and they complete TRU very fast. These players' HOD requests vary from low to average and they are labeled as *Veterans* as they are the most well performing group of players despite the high number of environment-related deaths. Likewise, cluster number 2 corresponds to players that die quite often mainly due to falling; it takes them quite a long time to complete the game; and they do not appear to ask for puzzle hints or answers. Players of this cluster are labeled as *Solvers*, because they are adept at solving the puzzles of TRU. Their long completion times, low number of deaths by enemies or environment effects indicate a slow-moving, careful style of play with the number one cause of death being falling (jumping).

Players of cluster number 3, form the largest group and are labeled as *Pacifists* as they die primarily from active opponents. The total number of their deaths varies a lot



(a) Cause of Death: Opponent, D_o

(b) Cause of Death: Environment, D_e

(c) Cause of Death: Falling, D_f

(d) Number of Deaths, D

(e) Completion Time, C

(f) Help on Demand, H

Fig. 6. The six component plane representations of the highest performing ESOM. The darker the area the higher the value of the component (feature).

but their completion times are below average and their help requests are minimal indicating a certain amount of skill at playing the game. Finally, the group of players corresponding to cluster number 4, namely the *Runners*, is characterized by players that die quite often and mainly by opponents and the environment. These players are very fast in completing the game (similar to the *Veterans*), while having a varying number of help requests which cover the majority of the H value range.

VII. Discussion & Conclusions

This paper provides an initial study on identifying different player types in a major commercial computer game via unsupervised learning. A large set of data obtained from 1365 players that completed the game Tomb Raider: Underworld is used for this purpose. Six statistical features corresponding to high-level playing characteristics of the core game mechanics are used as the inputs of an emergent self-organizing map. The input (feature) vector used consists of: 1) the game completion time; 2) the total number of deaths; 3) the total number of Help-on-Demand actions; and the proportion of deaths caused by the different causes of death in the game, categorized as 4) death by falling, 5) death due to the action of computer-controlled opponents, and 6) death due to hazards in the virtual environment. The highest-performing toroid-shaped ESOM trained on the data reveals four clusters of playing behavior — labeled as *Veterans*, *Solvers*, *Pacifists* and *Runners* — which are characterized by specific patterns of play, after the analysis of the U-matrix and its corresponding component planes. Importantly, the existence of four clusters of behavior, even in a fairly linear and restricted game like TRU, shows that players utilize the affordance space and flexibility offered by the design of the game, rather than simply using one specific strategy to get through the game. For example, the *Pacifists* are experts in terms of navigation and move rapidly through the virtual environment, but also respond badly to threats that are moveable or unexpected; whereas *Solvers* are excellent at solving puzzles, respond readily to moveable threats but die often from falling and are slow to complete the game.

When evaluating if the players of a game play as intended, or if unwanted or surprising (but unproblematic) behaviors occur, the type of results presented here are immediately useful. Importantly, the translation of player behaviors from raw data output to descriptions that take their basis in the game design and associated terminology is crucial in order for the information to be useful to the industry-based game designers.

The obvious step that will take the modeling of TRU players further is to extract additional features from the data available (e.g. use of different weapons and spatial/navigational behavior) and insert them in the input vector of the ESOM. Given a large feature set, automatic feature selection methods will most likely be employed to choose the most suitable feature subset that yields the highest performing ESOM. In addition to the high-level features, more data features corresponding to lower-level behaviors (e.g. number of times a weapon is fired in a specific level or sub-section of a game level, HoD-requests for specific in-game puzzles) will be extracted. The player types identified by the ESOM can then assist a machine learner trained on sequences of player actions over the various levels of the game using hidden Markov models or recurrent artificial neural networks. This work will be carried out on the current Tomb Raider game and any future installations in the series, in collaboration with Crystal Dynamics/EIDOS.

The proposed approach of player modeling through self-organization appears generic to any type of game genre, especially those featuring a central player character, as long as reliable playing behavior data is available; however, it should be noted that the features suitable for the investigation will vary between games. For example, shooting accuracy with different weapons appears as a suitable input feature of a shooter-type game. In general, the features chosen will depend on the core mechanics of the game as well as the overall purpose of the analysis in question.

The methodology can be used for automating, in part, the traditional exhaustive user and player testing procedures used in the games industry [2] by providing detailed, quantitative feedback on player behavior. This is plausible since the proposed approach provides the opportunity to evaluate in detail if a game design works as intended, by recognizing the patterns in how the game is played, and comparing this with the intentions of the design. Optimization of game design features — i.e. making sure that no feature of the game is under-used or misused — and *phenomenological debugging* — i.e. debugging of playing experience and game balancing — can also benefit from player behavior modeling. Furthermore, information about the different player types can be used during play to dynamically alter in-game controllable parameters (e.g. help on demand accessibility, difficulty of jumps) to adjust to the needs and skills of the player type identified in real-time and ensure variation in gameplay.

Acknowledgments

This work would not be possible without the game development companies involved. The authors would like to thank their colleagues at Crystal Dynamics and IO Interactive (IOI) for continued assistance with access to the EIDOS Metrics Suite and discussion of approaches, methods and results, including but certainly not limited to: Thomas Hagen and the rest of the EIDOS Online Development Team, Janus Rau Sørensen and the rest of the IOI User-Research Team, Tim Ward, Kim Krogh, Noah Hughes, Jim Blackhurst, Markus Friedl, Thomas Howalt, Anders Nielsen as well as the management of both companies. Special thanks also goes to Arnav Jhala for insightful discussions.

References

[1] K. Isbister and N. Schaffer, *Game Usability: Advancing the Player Experience*. Morgan Kaufman, 2008.

[2] J. H. Kim, D. V. Gunn, E. Schuh, B. C. Phillips, R. J. Pagulayan, and D. Wixon, "Tracking real-time user experience (true): A comprehensive instrumentation solution for complex systems," in *Proceedings of CHI*, Florence, Italy, 2008, pp. 443–451.

[3] A. Tychsen and A. Canossa, "Defining personas in games using metrics," in *Proceedings of Future Play 2008*. Toronto, Canada: ACM publishers, 2008, pp. 73–80.

[4] R. J. Pagulayan, K. Keeker, D. Wixon, R. L. Romero, and T. Fuller, *The HCI handbook*. Lawrence Erlbaum Associates, 2003, ch. User-centered design in games, pp. 883–906.

[5] A. Ultsch, *Kohonen Maps*, 1999, ch. Data Mining and Knowledge Discovery with Emergent Self-Organizing Feature Maps for Multivariate Time Series, pp. 33–46.

[6] T. Kohonen, *Self-Organizing Maps*. Springer, 2001.

[7] C. Thurau, C. Bauckhage, and G. Sagerer, "Combining self organizing maps and multilayer perceptrons to learn bot-behaviour for a commercial game," in *GAME-ON*, 2003, pp. 119–123.

[8] R. Houlette, *Player Modeling for Adaptive Games. AI Game Programming Wisdom II*. Charles River Media, Inc, 2004, pp. 557–566.

[9] D. Charles and M. Black, "Dynamic player modelling: A framework for player-centric digital games," in *Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education*, 2004, pp. 29–35.

[10] C. Thurau, C. Bauckhage, and G. Sagerer, "Learning human-like Movement Behavior for Computer Games," in *From Animals to Animats 8: Proceedings of the 8^{th} International Conference on Simulation of Adaptive Behavior (SAB-04)*, S. Schaal, A. Ijspeert, A. Billard, S. Vijayakumar, J. Hallam, and J.-A. Meyer, Eds. Santa Monica, LA, CA: The MIT Press, July 2004, pp. 315–323.

[11] C. Thurau, T. Paczian, and C. Bauckhage, "Is bayesian imitation learning the route to believable gamebots?" *International Journal of Intelligent Systems Technologies and Applications*, vol. 2, no. 2/3, pp. 284–295, 2007.

[12] R. Thawonmas, M. Kurashige, K. Iizuka, and M. Kantardzic, "Clustering of Online Game Users Based on Their Trails Using Self-organizing Map," in *Proceedings of Entertainment Computing - ICEC 2006*, 2006, pp. 366–369.

[13] G. N. Yannakakis and M. Maragoudakis, "Player modeling impact on player's entertainment in computer games," in *Proceedings of the 10^{th} International Conference on User Modeling; Lecture Notes in Computer Science*, vol. 3538. Edinburgh: Springer-Verlag, 24–30 July 2005, pp. 74–78.

[14] D. Thue, V. Bulitko, M. Spetch, and E. Wasylishen, "Interactive storytelling: A player modelling approach," in *The Third Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, Stanford, CA, 2007, pp. 43–48.

[15] C. White and D. Brogan, "The Self Organization of Context for Learning in Multiagent Games," in *Proceedings of the Second Artificial Intelligence and Digital Interactive Entertainment Conference (AIIDE)*. AAAI Press, 2006, pp. 92–97.

[16] R. Rosenthal, "Covert communication in laboratories, classrooms, and the truly real world," *Current Directions in Psychological Science*, vol. 12, no. 5, pp. 151–154, 2003.

[17] R. M. Gray, "Vector quantization," *IEEE ASSP Magazine*, pp. 4–29, 1984.

[18] A. Ultsch and L. Herrmann, "Architecture of emergent self-organizing maps to reduce projection errors," in *Proceedings of ESANN*, 2005, pp. 1–6.

[19] A. Ultsch and F. Moerchen, "Esom-maps: tools for clustering, visualization, and classification with emergent som," Dept. of Mathematics and Computer Science, University of Marburg, No. 46, Tech. Rep., 2005.

[20] J. H. J. Ward, "Hierarchical grouping to optimize an objective function," *Journal of the American Statistical Association*, vol. 58, pp. 236–244, 1963.

[21] A. Ultsch, "Maps for the Visualization of high-dimensional Data Spaces," in *Proceedings of the Workshop on Self-Organizing Maps*, 2003, pp. 225–230.