2017 IEEE 10th International Workshop on Computational Intelligence and Applications
November 11-12, 2017, Hiroshima, Japan pp. 171-176

Feature Extraction of Gameplays for Similarity Calculation in Gameplay Recommendation

Kazuki Mori*, Suguru Ito*, Tomohiro Harada†, Ruck Thawonmas† and Kyung-Joong Kim‡
*Graduate School of Information Science and Engineering
Ritsumeikan University, Kusatsu, Shiga, Japan
Email: {is0191kh, is0202iv}@ed.ritsumei.ac.jp
†College School of Information Science and Engineering
Ritsumeikan University, Kusatsu, Shiga, Japan
Email: harada@ci.ritsumei.ac.jp, ruck@is.ritsumei.ac.jp
‡Dept. of Computer Science and Engineering
Seoul, South Korea
kimkj@sejong.ac.kr

***Abstract*—This paper proposes a method for extraction of relevant features that represent a gameplay and are needed in gameplay recommendation. In our work, content based filtering (CBF) is adopted as the recommender algorithm. CBF exploits a heuristic that the user's previous ratings of items, gameplay clips in our case, can be used to derive the rating of an unrated similar item. In this work, in order to calculate the similarity between a pair of gameplays, a kind of autoencoder called Denoising Autoencoder is employed. Our experimental results confirm that the method can successfully extract features, based on which the resulting similarity between a pair of gameplays matches with their content and human perception.**

***Index Terms*—Procedural Play Generation; Recommender Systems; Denoising Autoencoder;**

I. Introduction

A gameplay clip is a video clip displaying the content of a game played by human or even AI players. At present, gameplay clips have been frequently uploaded or streamed to video sharing or streaming platforms such as YouTube or Twitch, some of which have around 10 million daily active users or spectators watching such clips. As a result, gameplay clips have been considered as promising new media, and recently Procedural Play Generation (PPG) [1] has been proposed with the objectives of automatically (procedurally) generating gameplays and recommending them to spectators. This work focuses on the recommender part of PPG.

A recommender is a system that selects and presents needed information, among a large amount of information, to users. Such systems are nowadays used in a variety of web sites such as the aforementioned video portal sites or social networking service sites. They can be divided into two categories: collaborative filtering (CF) [2], which is based on previous ratings by other users on items including an item of interest, and content based filtering (CBF) [3], which is based on the features of an item of interest, such as color or shape, and previous ratings of a targeted user to other items. Since one of the objectives of PPG is to procedurally generate a brand new gameplay, a recommender technique that fits this purpose is the one that can be performed for new items, or gameplays in our case, with no previous ratings. Hence, CBF that has an ability to do so is used in this work.

In CBF, in order to be able to calculate the similarity between a given pair of items, their features must be derived. Typically, this is done by, for example, manually tagging such an item by some keywords. However, this approach is not viable in PPG where gameplays are seamlessly generated by AI [4]. In this work, we, therefore, propose a method for automatically extracting relevant features from a gameplay.

II. Related Research

Typical existing work on video recommendation include Davidson et al. [5] and Gomez-Uribe & Hunt [6], which both aimed at creating a list of recommended videos to the users based on their viewing log. In the former work, the similarity between a pair of videos was defined by the number of times that they were both watched by users. The system recommends to a user of interest the top N videos that have the highest similarity to a seed video watched by the user. In the latter work, besides a conventional CF algorithm that recommends videos that have the highest predicted values, a number of other algorithms were used to increase the diversity in recommendation results, such as those considering seasonal factors (Christmas event etc.) and tag information. However, both of the aforementioned work did not consider the content of each video.

Sifa et al. [7] proposed a system for recommending video games. In their work, information on games that each user played and on time that each user spent to each game was used for deriving features on relations between games and

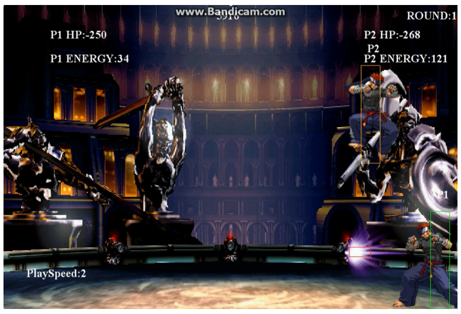

Fig. 1. A screenshot of FightingICE

users. These features were then used to predict the playing time that a user of interest will spend on each unplayed game which will be recommended to the user in decreasing order of predicted playing time. This kind of analysis is called player profiling [8], which can also be applied to player behavior prediction and cheating detection, etc. However, the work by Sifa et al. did not consider other factors, such as play contents, besides playing time.

Procedural Content Generation (PCG), compared to recently proposed PPG, is a more established and broader area focuses on automatic generation of game contents [9]. According to a recent definition [10], game contents targeted by PCG can cover a variety of components such as levels, maps, card textures, and stories. However, PCG typically does not either aim to generate a gameplay or target spectators. PCG's main targets are that of entertaining game players or that of assisting game developers while PPG targets spectators.

In regard to recommendation of gameplay clips, there exist previous studies [11][12]. In those studies, StarCraft, a real-time strategy game, was targeted. In the former study, game features were defined by exploiting the game domain knowledge, resulting in not only global features, such as the total number of actions (game events) by the player, but also local features that store information on the first timestamp of each unit production or building construction. In the latter study, Restricted Boltzmann Machine (RBM) was used in feature extraction, where the input images of RBM were reconstructed from a variety of information about units and buildings from replays at a timing of interest. Our present work is similar to [11] in that the game domain knowledge is exploited to determine which information should be used in defining the game state. However, in this work, a deep learning network is applied to such information directly, rather than to reconstructed images as done in [12], for extracting relevant information that represents a whole gameplay.

III. FightingICE

Although the proposed feature extraction method can be applied to any games, at the current stage of this research, we use a 2d-action-fighting game called FightingICE (Fig. 1) [13]. This game has been used as a platform for a game AI competition called the Fighting Game AI Competition, held at IEEE Conference on Computational Intelligence and Games since 2014. In this game, a round lasts 60s and consists of 3600 frames. Until the 2016 competition, each character has an initial health point (HP) of 0, and this value will be decreased upon receiving an attack by the opponent. In the end of a round, the character with a higher value of the remaining HP wins the round.

FightingICE allows the AI developer to obtain the game-state information at every frame, such as the position and HP of each character. In order to make a challenging situation for AI research, a delay of 15 frames (around 0.25s) was introduced, roughly representing a delay in response time of human players. However, since this work focuses of recommenders, we remove this delay from the system, enabling us to precisely obtain the current game state at each frame.

IV. Game Information

As done in previous work [11], [12], we focus on game information at each frame and use it to construct features of a given gameplay. In particular, we use the information on each character as follows:

- HP: The value of the HP of each character.
- ENERGY: The current value of the energy of each character. A certain amount of energy is required to perform some attack actions. The energy of a given character increases when its attack hits the opponent or when an attack of the opponent hits it.
- Position: The X and Y coordinates of each character.
- Speed: The horizontal and vertical speeds of each character
- Action: A vector showing which action is currently being performed among all 56 actions available in FightingICE by each character. It is represented by a one-hot vector of 56 bits.
- State: A vector showing the current state of each character among AIR, CROUCH, DOWN, and STAND in FightingICE. It is represented by a one-hot vector of 4 bits.
- Positions of the 1st to 3rd Hadouken's projectiles: The current X and Y coordinates of the ith projectile of Hadouken where $i = 1, 2, 3$ in launched order by each character; if such a projectile does not exist, its coordinates will be filled by "0". Hadouken consumes some amount of energy to release a projectile which moves at a steady speed towards the targeted direction. If Hadouken is performed n times, n projectiles will be released, each of which lasts until a certain amount of time has reached or until it hits the opponent. According to the current specification of FightingICE, at most three projectiles of each character can be present on the screen.

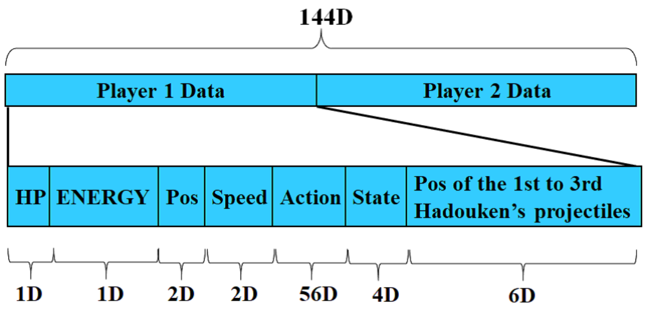

Fig. 2. A representation of the input-vector. Here "Pos" means position.

The above information is obtained for each character per frame. As a result, a vector of 144 dimensions is formed to represent the game state at a frame of interest (Fig.2).

V. Feature Extraction

Figure 3 depicts an outline of the proposed feature extraction method while Fig. 4 shows an architecture of the autoencoder in use. Since there are two 56D one-hot vectors among 144 dimensions of the input data per frame, there is a need to extract relevant features out of them. For this task, we use a kind of autoencoder called Denoising Autoencoder (DAE) [14]. Our DAE is composed of an input layer, a hidden layer, and an output layer, where the number of units in the hidden layer is set less than that of the input one. The objective for training DAE is that of minimizing the difference between the output and the input as given in Equation (1):

$$\min_{\mathbf{W},\mathbf{V},\mathbf{b},\boldsymbol{\mu}} \sum_{i=1}^{N} (\mathbf{x}_i - \widehat{\mathbf{x}}_i)^2 + \lambda(|\mathbf{W}|^{\mathbf{2}} + |\mathbf{V}|^{\mathbf{2}}) \tag{1}$$

$$\text{Encoder}: \mathbf{y} = f(\mathbf{W}\widetilde{\mathbf{x}} + \mathbf{b}) \tag{2}$$

$$\text{Decoder}: \widehat{\mathbf{x}} = g(\mathbf{V}\mathbf{y} + \boldsymbol{\mu}) \tag{3}$$

Here, N is the number of training frames for all considered gameplays, $\mathbf{x} \in \mathbb{R}^d$ is the input vector, $\mathbf{y} \in \mathbb{R}^k$ is the representation of $\mathbf{x}$ on lower-dimensional space, $\widehat{\mathbf{x}} \in \mathbb{R}^d$ is the output vector, resulting from reconstruction of $\mathbf{x}$ from lower dimensional $\mathbf{y}$, $\mathbf{W} \in \mathbb{R}^{k\times d}$ and $\mathbf{V} \in \mathbb{R}^{d\times k}$ are the weights from the input layer to the hidden layer and those from the hidden layer to the output layer, respectively. In addition, $\mathbf{b}$ and $\boldsymbol{\mu}$ are biases, $f(\cdot)$ and $g(\cdot)$ are the activation functions in use, and λ is a hyper parameter. For DAE, noises are added to $\mathbf{x}$, resulting in $\widetilde{\mathbf{x}}$ that will be the actual input to the network, see Equation (2), rather than $\mathbf{x}$ in the standard autoencoder.

After training, the output from the hidden layer of the trained DAE for each frame will be combined by mean pooling 3, resulting in a feature vector of 144 dimensions that represent a given gameplay.

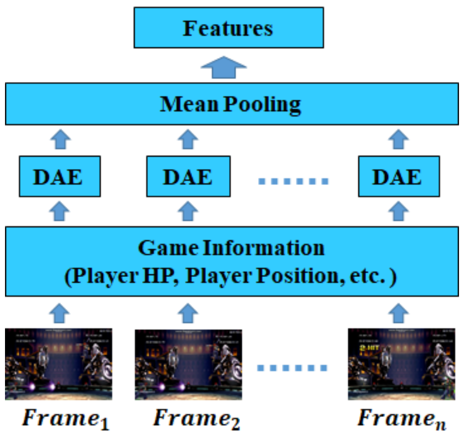

Fig. 3. A conceptual diagram of the proposed feature extraction method

VI. Experiments

We evaluate features extracted by the proposed method by examining whether the similarity, calculated based on the resulting features, between a pair of similar gameplays is high, or low for a pair of dissimilar ones.

A. Data Set

In our experiments, we use all AIs submitted to the 2016 Fighting Game AI Competition, excluding those that could not be run which results in 11 AIs. For these 11 AIs, we conduct a round-robin tournament, where each game is limited to only one round and 11 games played by the same AI at both sides are also included, and obtain 121 gameplays, resulting in $N = 435600$.

B. Data Normalization

We perform normalization to the elements corresponding to HP, ENERGY, Position, Speed, and Position of the 1st to 3rd Hadouken's projectiles in the input vector so that each of them has a mean of 0 and variance of 1 as follows:

$$x_{n_i} \leftarrow \frac{x_{n_i} - \bar{x}_n}{\sigma_n} \tag{4}$$

where n is the corresponding element in the input vector and $i = 1, 2, \ldots, 435600$.

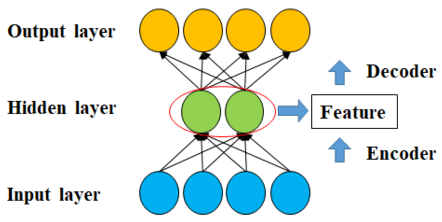

Fig. 4. An architecture of DAE

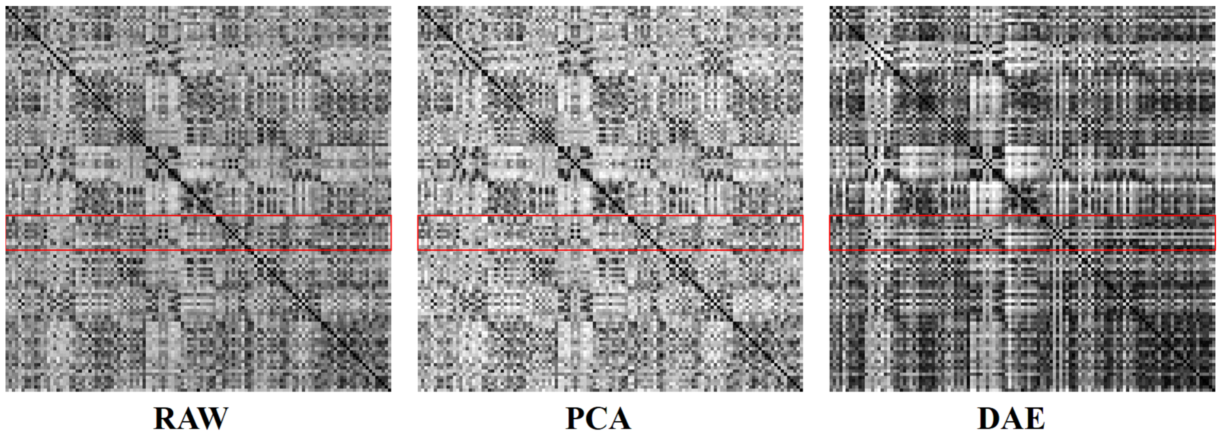

Fig. 5. Visualization results of the similarity between gameplays calculated based on features extracted by three methods

C. *Training of DAE*

All frames in each of the 121 gameplays are used for training the DAE. Masking noise is applied to each of the 144 input elements of DAE with a probability of 0.25. A sigmoid function and a linear function are used at the hidden layer and the output layer, respectively. The number of hidden units is set to 20 while that of both input and output layers is set to 144. All weights are initialized by a uniform distribution having the range of $[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}]$, where n is the number of units at the lower layer. Stochastic gradient descent is used as the optimizer. The learning rate α and the hyper parameter λ are both empirically set to 0.005. In addition, in order to alleviate the effect of weight initial values, we train DAE 10 times and use the average similarity among these 10 trials for each pair of gameplays.

D. *Experiment 1*

Here, we visualize the similarity between every pair of gameplays and compare three methods. The first one uses the mean of the 144D input vector over 3600 frames to represent a gameplay of interest (henceforth called **RAW**). The second one, **PCA**, uses principle component analysis to generate a 20D vector from each frame's 144D input vector and derives the mean vector to represent a game player of interest. The third one is the proposed method and is henceforth called **DAE**. Cosine similarity is used for calculating such similarity and the more the similarity is closer to 1 the darker color it becomes.

Figure 5 shows the visualization results of **RAW** (left), **PCA** (center) and **DAE** (right). Both rows and columns are indexed by the ID of a gameplay. All sub-figures have the darkest color on their diagonal, on which the similarity between the same content is visualized. However, **DAE** shows clearer color shades for other pairs of gameplays, for example, as shown in the area bounded by a red rectangle.

Figure 6 shows a zoomed version of the aforementioned area for each method where two white horizontal lines can be readily seen near the middle of **DAE**'s block, but not in the other blocks.Each line corresponds to a pair of gameplays in one of which both AIs stand at their initial position and repeat their action because they are implemented based on simple rulebase.In other words, each pair consists of a gameplay where both AIs move and fight and a gameplay where both AIs do not move. As a result, the similarity should be low for such a pair. Since such lines can not be observed in either **RAW**'s block or **PCA**'s block, it can be said that **DAE** extracts more relevant features than the other two methods.

E. *Experiment 2*

In this experiment, we examine if human perception on similar gameplays correlates with the cosine similarity based on features extracted by **DAE**. In particular, we conduct a user study where 13 participants, all college students with the age of 21 to 26 years, are each tasked with subjectively assessing the similarity of nine pairs of gameplay clips presented to them. The experiment's protocol in detail is as follows:

1) All gameplays where both characters do not move from their initial position are removed in advance. (Note that they are not removed in Experiment 1)
2) Each three of the nine pairs of gameplay clips in use are selected from the pairs of remaining gameplays that have the highest similarity (Highest), the least similarity (Least), and the similarity nearest to 0.5 (Middle), respectively.

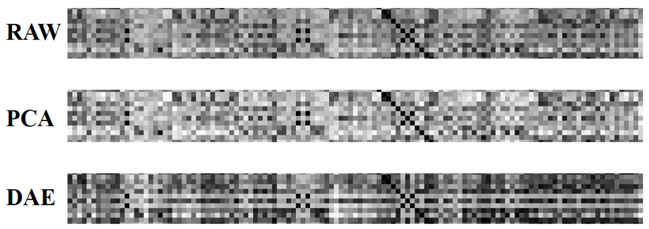

Fig. 6. Zoomed results of the area bounded by the red rectangle in Fig. 5

TABLE I
SPEARMAN'S RANK CORRELATION COEFFICIENT BETWEEN THE AVERAGE SCORE BY THE PARTICIPANTS AND THE COSINE SIMILARITY BASED ON FEATURES EXTRACTED BY **RAW**, **PCA**, AND **DAE**, RESPECTIVELY, WHERE p IS SHOWN IN THE PARENTHESES.

RAW	**PCA**	**DAE**
0.201 (0.604)	0.435 (0.242)	0.843 (0.004)

3) Since a round lasts 60s, to reduce a burden in watching gameplay clips, all selected gameplays are captured at 2X speed.
4) The selected nine pairs of gameplay clips are presented to each participant in a random order.
5) Each participant evaluates a given pair in a Likert scale as "Similar", "Somewhat similar", "Not very similar", and "Not similar", having the score of 4, 3, 2, and 1, respectively. During evaluation, the participant can arbitrarily replay any or both gameplay clips from any timing.

Figure 7 shows the resulting score for each category. It can be seen that the participants' ratings correspond to the similarity calculated based on the features extracted by **DAE**. We conduct a Kruskal-Wallis test against these categories and find that there is a significant difference between them ($p < 0.01$).

In addition, Table I shows Spearman's rank correlation coefficient between the average score by the participants and the cosine similarity based on features extracted by **RAW**, **PCA**, and **DAE**, respectively, for the above nine pairs. According to the results in this table, the association is statistically significant between the participants's average score and the **DAE**'s similarity, but not for the others.

In addition, we show a sequence of screen shots (at 5s, 15s, and 30s) for a pair having the least similarity and the highest similarity in Figs. 8 and 9, respectively. In the top row of Fig. 8, one can see that both characters (AIs) keep a certain distance and release Hadouken's projectiles to each other as long as they have enough energy while

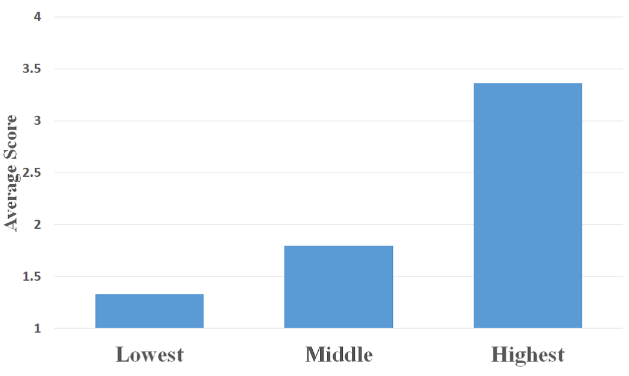

Fig. 7. Average score for each category of gameplay pairs by 13 participants

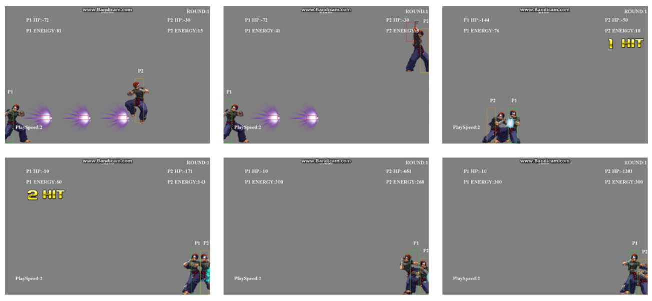

Fig. 8. Series of screenshots for two dissimilar gameplays[1,2]. For the sake of visibility, the grey background is used here.

both characters are close-range fighters in the bottom one. In Fig. 8, those screen shots indicate that all involving characters are close-range fighters.

A finding that we draw from the results is the proposed method can extract gameplay features, based on which the resulting similarity of a given pair of gameplays matches with human perception.

VII. Conclusions and Future Work

As a part of PPG, in order to be able to recommend gameplay clips to spectators, a method was proposed for extracting relevant features from a gameplay. From the results of two conducted experiments, it was confirmed that the method could successfully extract features, based on which the resulting similarity between a pair of gameplays matched with their content and human perception.

In the future, we will incorporate visual information from the game screen to the method and consider a way to handle temporal information residing in a sequence of frames forming a round. In addition, although a fighting game was considered in this work, we plan to extend the proposed method to other game genres, using, for example, research-oriented game platforms for Angry Birds [15] and Zelda [16].

Acknowledgment

This research was partially supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (2017R1A2B4002164) and by Strategic Research Foundation Grant-aided Project for Private Universities (S1511026), Japan.

References

[1] R. Thawonmas and T. Harada, "AI for Game Spectators: Rise of PPG," AAAI 2017 Workshop on What's next for AI in games, San Francisco, USA, pp. 1032–1033, Feb. 2017.
[2] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," Computer, 42(8), pp. 30–37, 2009.

[1]https://www.youtube.com/watch?v=hDu-zHBxz88
[2]https://www.youtube.com/watch?v=QwcVWW03yjQ
[3]https://www.youtube.com/watch?v=DPnYayZLuUw
[4]https://www.youtube.com/watch?v=J1vKNQUnTTs

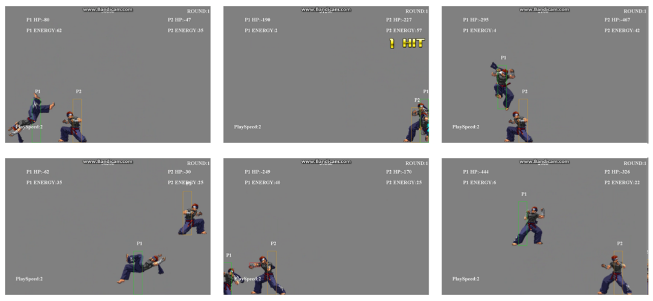

Fig. 9. Series of screenshots for two similar gameplays[3,4]

[3] M.J. Pazzani and D. Billsus, “Content-based recommendation systems,” The Adaptive Web, LNCS 4321, pp. 325–341, 2007.

[4] S. Ito, et al., “Procedural Play Generation According to Play Arcs Using Monte-Carlo Tree Search,” Accepted for presentation at the 18th annual European GAMEON Conference (GAMEON’2017), Carlow, Ireland, Sep. 2017.

[5] J. Davidson, et al., “The YouTube video recommendation system,” Proc. of the fourth ACM conference on Recommender systems (RecSys’10), Barcelona, Spain, pp. 293–296, Sep. 2010.

[6] C.A. Gomez-Uribe and N. Hunt, “The netflix recommender system: Algorithms, business value, and innovation,” ACM Transactions on Management Information Systems, 6(4), article no. 13, 2016.

[7] R. Sifa, C. Bauckhage, and A. Drachen, “Archetypal Game Recommender Systems,” Proc. of the 16th LWA Workshops: KDML, IR and FGWM, Aachen, Germany, pp. 45–56, Sep. 2014.

[8] R. Sifa, A. Drachen, and C. Bauckhage, “Profiling in Games: Understanding Behavior from Telemetry” Social Interaction in Virtual Worlds. Cambridge University Press, 2017. (in press)

[9] M. Hendrikx, S. Meijer, J. Van Der Velden, and A. Iosup. “Procedural content generation for games: A survey,” ACM Transactions on Multimedia Computing, Communications, and Applications, 9(1), article no. 1, 2013.

[10] A. Summerville, et al. “Procedural Content Generation via Machine Learning (PCGML),” arXiv:1702.00539, Feb. 2017.

[11] H.T. Kim and K.J. Kim, “Learning to recommend game contents for real-time strategy games,” Proc. of 2014 IEEE Conference on Computational Intelligence and Games (CIG 2014), Dortmund, Germany, Aug. 2014.

[12] H.T. Kim, Deep learning for game contents recommendation in real-time strategy games. Master’s thesis, Department of Computer Engineering, the Graduate School, Sejong University, Feb. 2015.

[13] http://www.ice.ci.ritsumei.ac.jp/ ftgaic/ (last accessed on August 10, 2017).

[14] P. Vincent, H. Larochelle, Y. Bengio, and P.A. Manzagol, “Extracting and composing robust features with denoising autoencoders,” Proc. of the 25th International Conference on Machine learning (ICML’08), Helsinki, Finland, pp. 1096–1103, Jul. 2008.

[15] https://aibirds.org/ (last accessed on August 10, 2017).

[16] N. Heijne and S. Bakkes, “Procedural Zelda: A PCG Environment for Player Experience Research,” Proc. of the 2017 International Conference on the Foundations of Digital Games (FDG’17), Hyannis, MA, USA, Aug. 2017.