Artificial Intelligence Based Game Levelling

Y. SARICA and M. CETIN*

***Abstract*— The applications of artificial intelligence (AI), which is a comprehensive information technology, have been closely related to game technologies. Today, artificial intelligence-based game development applications are increasing their popularity day by day. In this study, the levelling process of a 2-dimensional (2D) platform game has been investigated. The game developed and called "*Renga*" has a basic gameplay. Game data has been processed through an artificial neural network (ANN), *k*-nearest neighbour, decision and random tree algorithms and deep learning model that is trained with gameplay and user information. The classification process with the output data provides results for the next game level. In this way, the most effective playability impression that the developers offer to the game users has been created according to game. Furthermore, the variety of difficulty calculated with dynamic data by the user is provided by *Renga*, in which new sections/levels are created with user-specific assets. Thus, the most efficient gaming experience has been transferred to the users.**

Index Terms— **Artificial Intelligence, Difficulty Adjustment, Content Generation, k-Nearest Neighbor, Random Forest, Artificial Neural Networks.**

I. INTRODUCTION

AMONG the platforms, different types of games that can attract the attention of their target users are increasing and improving. Video games that contribute to this development are very successful in environmental design and transferring the realistic behavior of Non-Player Characters (NPC) [1]. In video games, which have become an important part of the entertainment industry, the goal is to optimize players' experience rather than creating the most difficult game [2-4]. A good video game has a long learning curve. Therefore, it is possible for the artificial intelligence-based-algorithm to update itself in accordance with the interaction of people who learn the game. Research on the content production of games helps to develop better quality or interesting games. The player's expectation is that the difficulty of the game matches his personal gaming skills [5, 6]. Therefore, a proper content generation depends on the algorithm producing a meaningful output with player performance. Depending on the algorithm, the difficulty level of the game should increase as the player's abilities develop [7].

Designing quality content for millions of players in the game industry has become the goal of game developers. In [8], a procedural level generator based on interactive evolutionary algorithm has been introduced for the platform game. Another automatic level generator based on Darwin's theory of Natural selection was proposed in [9]. In [10], a Rhythm-based approach was proposed for 2D Platform content generation. In [11], dynamic difficulty adjustment is provided by Polymorph approach which changes game difficulty depending on player performance. In the literature, there are various studies that are designed by machine learning (genetic algorithm, artificial neural network, support vector machine) [12-16], probabilistically techniques [17], Procedural Content Generation (PCG) [18-20] for dynamic game levelling. The difficulties caused by the behavior of NPC objects can be supported by artificial intelligence applications to maximize a user's enjoyment. The use of artificial intelligence in games is diversifying and deepening day by day. In [21], Yannanakis and Togelius have pointed out the following matters on the subject: behavior learning of non-player characters, exploration and planning applications, player modeling, artificial intelligence game competence applications, methodological content production, stories that may vary, the formation of realistic environmental movements, artificial intelligence game design, use of commercial games, general in-game artificial intelligence.

In artificial intelligence based studies proposed for dynamic game leveling, instead of randomly determining the course of the game, levelling is done taking into account the player's characteristics (aggression, courage, intelligence and cooperation) and the appropriate reaction is selected depending on these designs. Neural networks successfully applied to various games can be considered as a means of updating the artificial intelligence system. As an artificial intelligence system that can evolve over time, many game developers often do not use genetic algorithms because they need too much CPU power and are too slow to produce useful results [22]. Finite State Machines (FSMs) have often used by game developers because it is easy to implement, test, modify, and personalize [22]. Increasingly, there has been a tendency towards Fuzzy State Machines (FuSMs) since the use of fuzzy logic allows the recognition of non-binary conditions [22]. Demasi and Cruz used the genetic algorithm technique to protect the most appropriate agents according to the player's game performance [23]. In addition, there are several studies

YUNUS SARICA, is with Department of Computer Engineering, University of Pamukkale University, Denizli/Turkey,(e-mail: yunusssarica@gmail.com).
https://orcid.org/0000-0002-1969-9005

MERIC CETIN, is with Department of Computer Engineering, University of Pamukkale University, Denizli/Turkey,(e-mail: mcetin@pau.edu.tr).
https://orcid.org/0000-0002-7871-4850,
*Corresponding Author.

Manuscript received November 05, 2019; accepted April 04, 2020.
DOI: 10.17694/bajece.642973

based on Gaussian Mixture Module with dynamic scripting [24, 25].

In this study, game levels for *Renga*, which was developed as a 2D platform game, have been identified based on artificial intelligence. In this context, perception of difficulty in infinite games has been examined, balanced the player performance of *Renga* game and then different game levels have been designed according to player ability. As a result of processing the game data with artificial intelligence methods (artificial neural network, k-nearest neighbor, decision and random tree algorithms, deep learning model), it is provided to produce game-specific procedurally appropriate content and meaningful outputs specific to the player. Algorithmically variable game content has been generated by PCG, which significantly has reduced the time and cost of development processes for game developers. In this way, the most effective playability impression that the developers offer to the game users has been created according to *Renga*. Furthermore, the variety of difficulty calculated with dynamic data by the user is provided by *Renga*, in which new sections/levels are created with user-specific assets. Thus, the most efficient gaming experience has been transferred to the users.

II. PLATFORM GAMES AND *RENGA*

In this section, the process and technological developments of platform games that have been on the agenda since the beginning of video game history are presented. Platform games are the first phase of video game history. Bertie the Brain is known as the first industrial game developed in the 1950s. This process continued eight years later with William Higinbotham's Tennis for Two. Pong, as the first realization of this game, has become one of the platform games [26]. Then, game technologies have developed into different categories. Productions such as Super Mario, Contra and Metal Slug have not lost their popularity despite the emergence of 3D games, and then, a new era of video games began with platform games such as Limbo, Inside, Ori and the Blind Forest. Many of these games have a special place today because they are widely distributed and can make their voices heard through digital gaming platforms [27].

After all these similarities and examples that have continued its genre, it was decided to perform this study with platform games. In the comparative tests, it has been paid attention that the developed game (*Renga*) can work in harmony with the selected artificial intelligence method.

Considering the existence of common objects, platform games can generally consist of a character, enemy, interactive objects and various platform groups. Although the aim of the selected character is to terminate the game parts, it is also important to be able to produce the appropriate level according to the game performance. One of the goals of designed game is to help the developer to design the level by providing sufficient data as soon as possible while building the sections. In this study, a 2D platform game (*Renga*) is designed which is simple to use and diversifies game assets. The developed game has been also introduced globally to the literature. In *Renga* methodological content production is based on an artificial intelligence algorithm and the identification data obtained in this way contribute to the change of in-game dynamics. In addition, a new data set was created by processing the data collected over the game. Game design is based on basic steps for artificial intelligence methods used in the study. The degrees of difficulty in the game depend on the numerical properties of the game assets. The main characteristics of the interaction of this process are:

- Simple gameplay,
- Interaction between assets,
- Updating the character structure,
- Contribution to the game score
- Academic study
- To provide entertainment to the global users

Renga, which is compatible with phones with Android operating system, is developed by Unity platform. *Renga* is an infinite game that can be played until the player makes a mistake. The game engine Unity is widely used in such games because it supports 3D modeling and animation design as well as 2D game development [28]. Each field in *Renga* consists of a scene whose content is full of game objects (“sprite”, “texture”, “prefab” and “object”). The game has 4 scenes actively: “Menu”, “howToPlay”, “gameplay” and “end” scenes. In the “menu” scene, the data to be tested is collected. Users have access to the code file of any scene in Unity through information on randomness value, horizontal speed, gravity, jump value and distance between obstacles. The game process takes place on the "gameplay" scene. A gameplay scene for the *Renga* is illustrated in Fig.1. Only *Renga* is available as game asset in this scene. Background, floor, “Aheng” and “Rengec” are dynamically composed.

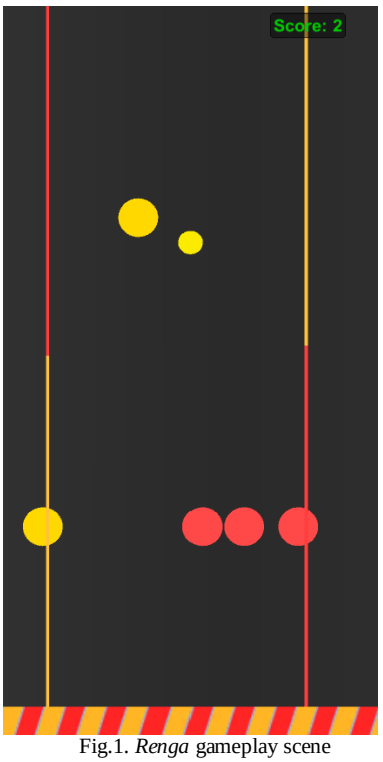

Fig.1. *Renga* gameplay scene

III. ARTIFICIAL INTELLIGENCE BASED GAME LEVELING

In this section, the effects of *Renga* game for the user, how the data is collected and the results obtained with the technology and methods used in this process are explained.

A. Data Collection

In order to control the game, the collected data is used to create a level, which affects the type and difficulty of the level. The statistical characteristics such as how often the player jumped, died, how much he caught *Renga* cannot be directly controlled by the game because of they depend on the player's skill and playing style. In the tests, the game levels for a particular player have been dynamically generated as:

1. A first level is generated with random parameters.
2. Game features are recorded.
3. Using recorded game features, a new adapted level is generated based on player experience.

The tests in this study are based on the approximately 1500 game session played by the 15 player (each user played the game at least 5 times with a randomly generated initial value). In the simulations, 80% of the input data set was used as training data. Sample game feature records of any user according to random parameters (seed, speed, gravity, jump, distance) are given in Table I.

TABLE I
RANDOM GAME SAMPLES FOR ANY USER

PHONE_ID	P1	P2	P3	P4	P5	P6	P7
ABCDEF9876	50	1.0	12	4.0	1.8	0	0
ABCDEF9876	50	1.0	10	4.0	1.8	0	0
ABCDEF9876	50	1.0	14	4.0	1.8	0	0
ABCDEF9876	50	1.0	12	5.5	1.8	0	0
ABCDEF9876	50	1.0	12	2.5	1.8	0	0

*P_1= seed value, P_2= speed value, P_3= gravity value, P_4= jump value, P_5= distance value, P_6= static value, P_7= score.

The "PlayerPrefs" feature provides access to game information within Unity. This data, which was added in the database before, was obtained with datas.php file. The web service is provided with the WWWForm() object specified in Unity. The data is then transferred to the server, "http://gameonyou.tk/". *Renga* is available to users in the Google Play Store without academic features. Therefore, the data is primarily hosted on the local server created with xampp. In addition, the application file for the academic version externally is shared with users via the address, "http://gameonyou.tk/Renga.apk". In the file dump named dbConnData.php, records are processed by connecting to the database with the pdo structure.

First of all, when the user runs the game for the first time, the in-game data is collected and inserted to the database. Gameplay data, except for fixed values, were examined by increasing or decreasing each value at a certain rate, except for a general data type. If any row given in Table 1 is considered; as the values changed, the other values remained constant and this situation continued with combinations. These records are explained in detail as follows:

Seed: Seed is a fixed value and set to 50. All users' test data were used with this seed value in artificial intelligence processes. Users who have completed all test values will start playing different sections with a different seed value.

Speed: The default value of 1.0 is the feature that determines the speed of *Renga* during gameplay. With the values of 0.8 and 1.2, it was aimed to observe the gameplay status of the users. Reaching the Rengecs is necessary for the advancement of *Renga*.

Gravity: The default value of 10.0 allows vertical movement of *Renga* in a constant direction during gameplay. Interactions of 12.0 and 8.0 values in users were examined. If *Renga* hits the ground due to this value, the game will end. The user protects *Renga* from the ground by touching the screen during the gameplay.

Distance: The default is 1.6. This variable specifies the value of the distance between the rails. It is differentiated with 1.0 and 3.0 values. It is inversely proportional to the speed value and can change the perception of difficulty.

Jump: The default value is 4. The effect of the differences with the binary values of 5.5 and 2.5 was observed in the users. The user can increase the height of the vertical moving *Renga* by the magnitude of this value.

The importance of jump and gravity values was considered when passing over or under the distinction points of Rengecs. Distinction points where *Renga* cannot pass are prevented from occurring. When enough data (recorded at 5 different times) is collected from the user, the in-game values will change. After all these collected data, the results of the related user are classified as easy, medium or difficult in designed artificial intelligence models. Since the initial data collected from the user changes after the new level proposed by the artificial intelligence model, the user will be able to continue to play the game more efficiently.

B. Examination of Renga Data

The data collected by the *Renga* game is sufficient for user-based difficulty control and adjustment. According to the input set given as an example in Table 2, it will be appropriate to mention the data that creates the algorithm inputs and the features of these data before explaining the artificial intelligence model to be created.

Randomness: It can be defined as the renovation value. All assets in the game are created according to a random value. The user plays at least 5 games for each data of the same randomness value. According to Eq. (1), it is possible for a user to present a minimum of 120 result outputs in the test environment through the designed artificial intelligence models.

$4!=24 \rightarrow 24*5=120$ (1)

In this way, new data can be created for difficulty control using the in-game values in the different seed values.

TABLE II
SAMPLE GAME DATA

PHONE_ID	K1	K2	K3	K4	K5	K6
ABCDEF123	42	1.5	7	2.5	3	6
ABCDEF123	42	1.5	7	2.5	3	9
ABCDEF123	42	1.5	7	2.5	3	9
ABCDEF123	42	1.5	7	2.5	3	10
ABCDEF123	42	1.5	7	2.5	3	4

*K_1= randomness, K_2= horizontal speed, K_3= gravity value, K_4= jump value, K_5= distance between obstacles, K_6= score.

Horizontal speed: The path taken from the beginning of the game by the "K_2' values in Table 2. According to this table;

$K_2 = 0{,}016*1{,}5=0{,}024$ (2)

In Eq. (2), 0.024 units of horizontal position changes occur for each frame per second. Increasing horizontal speed, which is one of the inputs in the artificial intelligence model, will increase the perception of difficulty in the game cause the user to be in easy class.

Gravity: It is vertical movement value, created with the "Rigidbody" feature, assigned to the *Renga* object from the beginning of the game. According to the gravity values given in Table 2 and equation (3), 0.112 unit changes occur for each frame.

$K_3 = 0{,}016*7 =0{,}112$ (3)

Increasing this value is inversely proportional to the jump value. There should be a balance between these values. Increasing this value is inversely proportional to the jump value. Increasing the value increases the difficulty, and the fact that it is too low makes the game impossible.

Jump: This variable value of action which user interacts with. Each time the user touches the screen, *Renga* will move vertically upwards by 2.5 units. During this movement, gravity also continues the downward movement. The accelerated *Renga* starts to move down again vertically. If the value is too high, even the lowest gain will lead to an impossible game.

Distance between obstacles: It is the distance unit between Rengecs. There is a 3-unit distance between the Rengecs, which are proportioned by two color segments (yellow and red) according to their seed value. Increasing this value makes the game easier, while decreasing it can significantly differentiate the perception of difficulty in the user experience.

Score: This value determines the player's highest gain in a game process. It also represents the number of Rengec that could be passed. It is the most important factor in the classification of difficulties for users. These data are differentiated in the results section.

During the *Renga* design, many different technical fields have been used besides the package programs. In the following sections, used artificial intelligence algorithms and findings are presented.

C. Methods

In this study, the recursively used data was tested in various artificial intelligence models such as k-nearest neighbor, random forest, deep learning and artificial neural networks in the category of classification algorithms and analyzed through confusion matrix. Success and error coefficients were determined based on the variables within each method and were detailed for future studies.

1) k-Nearest Neighbor

The *k*-nearest neighbor method performs the classification process based on similarities where the data is clustered in cartesian plane. According to the selected value of *k*, which cluster the data belongs to is determined [29]. The pseudo-code expression for this algorithm is given as follows:

*% **X:** training data, **Y:** class labels of **X**, x: unknown sample*
- ***Determine** parameter **k**: the number of* nearest neighbor
- ***Classify** (**X**, **Y**, x)*
- ***for** i = 1 to **m do***
  - ***Compute** distances **d**(X_I,x) between the query-instance and all training data.*
- ***end for***
- ***Compute** set I containing indices for the **k** smallest distances **d**(X_I,x).*
- ***return** majority of the category label for {Y_I}*

2) Random Forest

Random forests, which are commonly used in classification or regression processes, consist of a large number of individual decision trees [30]. Random forest is an efficient tree algorithm that can model game data collected from different users in a compatible way. The pseudo-code for this algorithm is given as follows [31]:

- ***Generate c** classifiers*
- ***for** i = 1 to **c do***
  - *Randomly sample the training data D with replacement to produce D_i*
  - ***Create** a root node, N_i containing D_i.*
  - ***Call BuildTree**(N_i)*
- ***end for***

BuildTree(N):
- ***if** N contains instances of only one class **then***
  - ***return***
- ***else***

- *Randomly* ***select*** *x*
- ***Select*** *the feature F with the highest information gain to split on*
- ***Create*** *f child nodes of N , N_1,...,N_f , where F has f possible values (F_1, ...,F_f)*
- ***for*** *i = 1 to* ***f do***
  - ***Set*** *the contents of N_i to D_i, where D_i is all instances in N that match*
  - F_i
  - ***Call BuildTree****(Ni)*
- ***end for***

➢ ***end if***

3) Deep Learning

Deep learning is a subfield of machine learning framework that involves one or more layered artificial neural networks. The difference from artificial neural networks is the ability to perform more operations as given in [32, 33]. In this study, deep learning algorithms were used to see the contribution of newly added game-data during training.

4) Artificial Neural Network

Artificial Neural Network is a computational model inspired by biological neural networks. An artificial neural network may comprise a plurality of neurons arranged in a series of layers. The input layer receives various forms of information. These data are passed through one or more hidden layers and converted to the output unit where the input can be used. An example of pseudo-code for back-propagation algorithm in training ANN is given as [34]:

*% **X:** training data of size mxn, **y:** class labels of **X**, w: the weights for respective layers, l: the number of layers*

*% $\boldsymbol{D^{(l)}_{ij}}$ the error for all **l, i, j,** $\boldsymbol{t^{(l)}_{ij}}$ = 0, for all **l, i, j.***

➢ ***for*** *i = 1 to* ***m do***

- $\boldsymbol{a^l = feedforward(x^{(i)}, w)}$
- $\boldsymbol{d^l = a(L)-y(i)}$
- $\boldsymbol{t^{(l)}_{ij} = t^{(l)}_{ij} + a^l_j * t^{(l+1)}_i}$

➢ ***if*** $j \neq 0$ ***then***

- $\boldsymbol{D^{(l)}_{ij} = (1/m)* t^{(l)}_{ij} + \lambda w^{(l)}_{ij}}$

➢ ***else***

- $\boldsymbol{D^{(l)}_{ij}}$ = *(1/m)** $\boldsymbol{t^{(l)}_{ij}}$, *where* $(d/d\boldsymbol{w^{(l)}_{ij}})*j(w)= \boldsymbol{D^{(l)}_{ij}}$

➢ ***end if***

➢ ***end for***

IV. Simulation Results

In this study, it is assumed that all users have played the game for the first time. The data set used for *Renga* consists of data from multiple users. Although the values for the application are the same, differences in points affect the result. The main reason is that the records are classified as *hard*, *medium* and *easy* according to the data content called "score". As a result of the tests, it was found that the most important factors affecting the score were gravity and speed variables. In the experimental tests, the simulations were tried on the artificial intelligence methods mentioned above with many different parameters and the most successful results were recorded. The results provide the level identification information for the *Renga* game. This data can be used to provide a more efficient level and in-game dynamics to the players. In the tables below, the best performance rates obtained for artificial intelligence methods are given.

TABLE III
PERFORMANCE RATES IN DEEP LEARNING

	Hard	**Medium**	**Easy**	**Consistency**
Hard *(Estimated)*	9	20	0	% 31.03
Medium *(Estimated)*	12	9	10	% 29.03
Easy *(Estimated)*	92	151	345	% 58.67
Singular Performance	% 7.96	% 5.00	% 97.18	

In deep learning tests, data has been trained according to the model of 2 hidden layer network. The best results for the tests with different activation functions were obtained in *maxout* function. The performance of the deep learning algorithm according to the learning rate (0.6) in the test data is %56.02.

TABLE IV
PERFORMANCE RATES IN *k*-NEAREST NEIGHBOR

	Hard	**Medium**	**Easy**	**Consistency**
Hard *(Estimated)*	42	50	0	% 45.65
Medium *(Estimated)*	78	154	0	% 66.38
Easy *(Estimated)*	0	0	0	% 0.0
Singular Performance	% 35.00	% 75.49	% 0.0	

Many tests have been performed with different k values for the k-nearest neighbor method which classifies according to similarities in the data. According to the simulation results, the best performance of the algorithm with the related learning rate was obtained for k = 5. The performance of the k-nearest neighbor algorithm according to the learning rate (0.8) in the test data is %60.49. It is generally considered that the k-nn method should not be used in inconsistent data because it has different gains for variable user types.

TABLE V
PERFORMANCE RATES IN RANDOM FOREST

	Hard	**Medium**	**Easy**	**Consistency**
Hard *(Estimated)*	4	1	0	% 80.00
Medium *(Estimated)*	13	40	17	% 57.14
Easy *(Estimated)*	27	48	174	% 69.88
Singular Performance	% 9.09	% 44.94	% 91.10	

As shown in Table 5, the change of algorithmic parameters for random forest did not bring an effective difference to the results. For this reason, the samples were differentiated and the findings were separated according to linear-complex samples. The performance of the random forest algorithm according to the learning rate (0.8) in the test data is %63.89.

The process that leads to the measurement of the level of the game allows the new values to be generated by the obtained values to be tested on the same users and thus to determine the new perception of difficulty. The decision tree structures in the random forest algorithm according to gravity and speed variables, which is one of the most important factors affecting the score, are presented in Figure 2 and Figure 3.

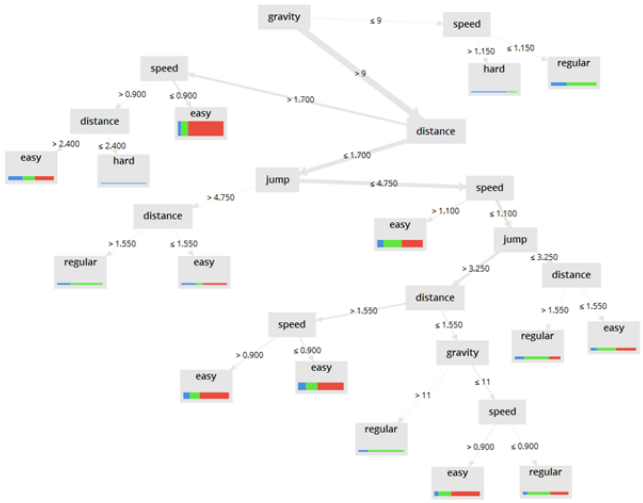

Fig.2. Decision tree method via gravity

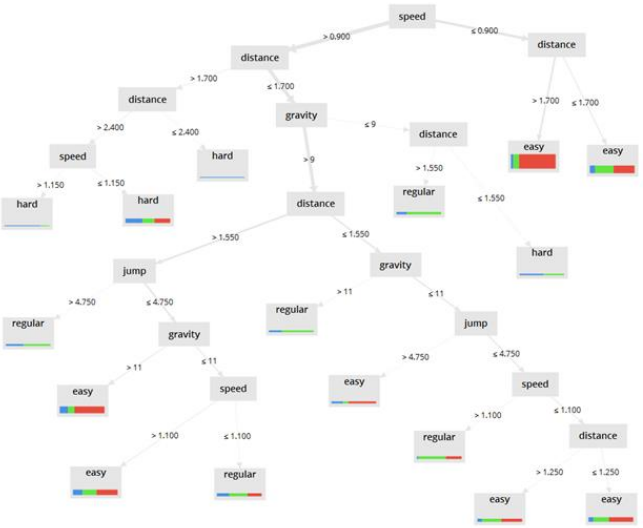

Fig.3. Decision tree method via horizontal speed

TABLE VI
PERFORMANCE RATES IN ARTIFICIAL NEURAL NETWORK

	Hard	Medium	Easy	Consistency
Hard *(Estimated)*	0	0	0	% 0.0
Medium *(Estimated)*	15	28	12	% 50.91
Easy *(Estimated)*	29	61	179	% 66.54
Singular Performance	% 0.00	% 31.46	%93.72	

The performance of the ANN algorithm according to the learning rate (0.8) in the test data is %63.58. In the experimental simulations, one hidden layer, one input layer and one output layer have been used for the ANN algorithm. The best results for the tests with different activation functions were obtained in sigmoid function for ANN implementation. According to the tests, ANN method is the second most efficient algorithm. However, it was found that the hard tag could not be predicted by the data sets mentioned.

V. CONCLUSION

In this paper, a level identification procedure has been performed with different methods and different parameters. This classification process has been tested on the users of *Renga*, an infinite game. The proposed structure provides an online gaming adaptation mechanism that can be used to effectively optimize the player experience. It is seen that the best method used in the study is the random forest algorithm. Furthermore, the random forest algorithm provided the best classification distribution. Besides the success rate, it is seen that it constitutes an efficient method for classification. The models of decision trees it offers are guiding and can provide high benefits. In a data set classification in which users' game identities are included, it is anticipated that new studies may emerge. In this way, the game can be played over certain basic users with certain values and data collection can make the leveling work more efficient. According to the results, the in-game values offered to the users gave positive results and it was observed that the game pleasure of the users was satisfactory. A short way have been shown to developers to present this experience, which saves time for improvements. The evaluations that can be made through this game can give an idea to be applied to other platform games. In the later stages of the studies, it was paved the way for its application in platform games with different genres and rich content.

ACKNOWLEDGMENT

This study was supported by Scientific Research Coordination Unit of Pamukkale University under the project number 2018FEBE003.

REFERENCES

[1] Y. Sarica “Game Levelling with Artificial Intelligence.” Master Degree Thesis, Pamukkale University, The Graduate School of Natural and Applied Science, 2019

[2] A. J. Baldwin. “Balancing act: the effect of dynamic difficulty adjustment in competitive multiplayer video games”, 2016.

[3] Y. Zhang, S. He, J. Wang, Y. Gao, J. Yang, X. Yu, L. Sha. “Optimizing player's satisfaction through DDA of game AI by UCT for the Game Dead-End”. In Natural Computation, Sixth International Conference on,Vol. 8, 2010, pp. 4161-4165.

[4] J. P. Gee. “What video games have to teach us about learning and literacy”. Computers in Entertainment, 1(1), 2003, 20-20.

[5] M. Csikszentmihalyi. “Flow and the psychology of discovery and invention”. Harper Perennial, New York, 1997, 39.

[6] R. Hunicke. “The case for dynamic difficulty adjustment in games”. In Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology. 2005, pp. 429-433.

[7] J. Sinclair. "Feedback control for exergames". Theses: Doctorates and Masters, 2011

[8] M. Kerssemakers, J. Tuxen, J. Togelius, G. N. Yannakakis. “A procedural procedural level generator generator”. In 2012 IEEE

Conference on Computational Intelligence and Games, 2012, pp. 335-341.
[9] F. Mourato, M. P. dos Santos, F. Birra. "Automatic level generation for platform videogames using genetic algorithms". In Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology, 2011, p. 8.
[10] G. Smith, M. Treanor, J. Whitehead, M. Mateas, (). Rhythm-based level generation for 2D platformers. In Proceedings of the 4th International Conference on Foundations of Digital Games, 2009, pp. 175-182).
[11] M. Jennings-Teats, G. Smith, N. Wardrip-Fruin. "Polymorph: dynamic difficulty adjustment through level generation". In Proceedings of the 2010 Workshop on Procedural Content Generation in Games 2010, p. 11.
[12] F. Mourato, M. P. dos Santos, F. Birra. "Automatic level generation for platform videogames using genetic algorithms". In Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology 2011, p. 8
[13] L. Ferreira, C. Toledo. "A search-based approach for generating angry birds levels". In Computational intelligence and games, 2014.
[14] L. Galway, D. Charles, M. Black. "Machine learning in digital games: a survey". Artificial Intelligence Review, 29(2), 2008, 123-161.
[15] P. Spronck, I. Sprinkhuizen-Kuyper, E. Postma. "Online adaptation of game opponent AI in simulation and in practice". In Proceedings of the 4th International Conference on Intelligent Games and Simulation, 2003, pp. 93-100.
[16] D. Johnson, J. Wiles. "Computer games with intelligence". In Fuzzy Systems. The 10th IEEE International Conference on, Vol. 3, 2001, pp. 1355-1358.
[17] M. Persson. "Infinite Mario bros". 2008, Online Game.
[18] W. Baghdadi, F. S. Eddin, R. Al-Omari, Z. Alhalawani, M. Shaker, N. Shaker. "A procedural method for automatic generation of spelunky levels". In European Conference on the Applications of Evolutionary Computation, 2015, pp. 305-317.
[19] G. Smith, M. Treanor, J. Whitehead, M. Mateas. "Rhythm-based level generation for 2D platformers". In Proceedings of the 4th International Conference on Foundations of Digital Games, 2009, pp. 175-182.
[20] V. der Linden, R. R. Lopes, R. Bidarra, "Designing procedurally generated levels", In Proceedings of the second workshop on Artificial Intelligence in the Game Design Process, 2013.
[21] G. N. Yannakakis, J. Togelius. "A panorama of artificial and computational intelligence in games". IEEE Transactions on Computational Intelligence and AI in Games, 7(4), 2014, 317-335.
[22] S. Woodcock, J. E. Laird, D. Pottinger, "Game AI: The state of the industry". Game Developer Magazine, 8,c2000.
[23] P. Spronck, I. Sprinkhuizen-Kuyper, E. Postma. "Difficulty scaling of game AI". In Proceedings of the 5th International Conference on Intelligent Games and Simulation, 2004, pp. 33-37.
[24] S. Lee, K. Jung. "Dynamic game level design using gaussian mixture model". In Pacific Rim International Conference on Artificial Intelligence, 2006, pp. 955-959.
[25] P. Spronck, M. Ponsen, I. Sprinkhuizen-Kuyper, E. Postma. "Adaptive game AI with dynamic scripting". Machine Learning, 63(3), 2006, 217-248.
[26] S. L. Kent. "The Ultimate History of Video Games: From Pong to Pokemon-The Story Behind the Craze That Touched Our Lives and Changed the World", 2001, New York: Three Rivers Press.
[27] J. Togelius, S. Karakovskiy, J. Koutník, J. Schmidhuber. "Super Mario Evolution. In Computational Intelligence and Games", IEEE Symposium, CIG 2009, pp. 156-161.
[28] J. K. Haas. "A History of the Unity Game Engine", 2014.
[29] B. Tay, J. K. Hyun, S. Oh. "A machine learning approach for specification of spinal cord injuries using fractional anisotropy values obtained from diffusion tensor images". Computational and mathematical methods in medicine, 2014.
[30] L. Breiman. "Random forests". Machine learning, 45(1), 2001, 5-32.
[31] N. Sirikulviriya, S. Sinthupinyo. "Integration of rules from a random forest". In International Conference on Information and Electronics Engineering, Vol. 6, 2011pp. 194-198.
[32] Y. LeCun, Y. Bengio, G. Hinton. "Deep learning". Nature, 521(7553), 2015, 436-444.
[33] X. Yao. "Evolving artificial neural networks". Proceedings of the IEEE, 87(9), 1999, 1423-1447.
[34] H. Guo, H. Nguyen, D. A. Vu, X. N. Bui. "Forecasting mining capital cost for open-pit mining projects based on artificial neural network approach". Resources Policy, 101474, 2019.

BIOGRAPHIES

YUNUS SARICA was born in Van, Turkey, in 1993. He received the B.S. and M.S. degrees in computer engineering from the Pamukkale University, Denizli. Since 2015, he has been working as a computer engineer. Since 2015, he had been a Research Assistant with the Computer Engineering Department, Pamukkale University, Denizli, through the M.S. degree. His research interests include game programming, artificial intelligence, desktop and web application programming. Beside these, he is interested in amateur theatre acting, voice-over and contemporary visual technologies.

MERIC CETIN received B.Sc in Electrical & Electronics Engineering from Pamukkale University in 2003; M.Sc. in Electrical & Electronics Engineering from Institute of Natural Sciences, Pamukkale University, in 2006. She obtained Ph.D. degree in Electrical & Electronics Engineering from Pamukkale University in 2015. She is currently an Assistant Professor at the Computer Engineering Department, Pamukkale University. Her research interests are in model predictive control, machine learning, and adaptive control with computational intelligence techniques. Dr. Cetin is a member of the European Embedded Control Institute and reviewer of several international journals.