

DETECTION OF PLAYER LEARNING CURVE IN A CAR DRIVING GAME

Boyan Bontchev¹, Dessislava Vassileva²

¹*Dep. of Software Technologies, Faculty of Mathematics and Informatics, Sofia University (BULGARIA)*

²*Scientific Research Centre (NIS) at Sofia University (BULGARIA)*

Abstract

Detection of learning curves of player metrics is very important for the serious (or so called applied) games, because it provides an indicator representing how players master the game tasks by acquiring cognitive abilities, knowledge, and necessary skills for solving the game challenges. Real time identification of specific patterns in the learning curve of a particular player may be applied for dynamic adjusting of learning task difficulty and audio-visual features of the game. The paper presents a method for automatic and straightforward detection of specific learning curves at run time within a 3D video game of car driving in various weather conditions. The method uses a client-side software component called "Player-centric rule-and-pattern-based adaptation asset" and developed in the scope of the RAGE (Realising and Applied Gaming Ecosystem) H2020 project. This component is integrated within the video game in order to detect dynamically specific player learning curve patterns of moving average of overall player performance. The paper explains how these patterns were identified while players play the game and, next, how are defined formally within the asset for a detection at run time. The presented results show that the patterns are found in consecutive game sessions, although the changes in player performance and playing time.

Keywords: Player learning curve, detection, pattern, video game, component, RAGE.

1 INTRODUCTION

Today, video games are present in the lives of people of all ages and their use has increased steadily overtime. Already they are part of our culture and they have become one of the most common and popular forms of entertainment. Video games are engaging, inspired and can capture and hold the attention of players for hours [1]. Presently, video games are much more than just fun. They are used by psychologists, educators, neuroscientists and marketing specialist in order to enhance learning, to attract attention, for psychological therapy, for investigating of the brain behaviour, etc. [2], [3], [4]. For video games used with purpose different from pure entertainment, Sawyer and Rejeski introduce the term *serious games* [5]. Through serious games people can acquire new knowledge and skills. Moreover, they have a positive impact on the areas related to cognitive, motivational, emotional and social issues [6].

In this article, we focus on serious games used for learning objective and we present an approach for detecting of player learning curves used in a car driving game. A learning curve (called also experience curve, or productivity curve) provides a graphical representation of progress in learning over time [7]. In video games, the learning curve illustrates how players must spend time for mastering the game by both developing playing skills and acquiring cognitive abilities and knowledge for solving the game challenges [8]. Naturally, learning curves vary from game to game and, as well, from player to player. A player learning curve represents the progress in learning and, hence, performance specific to an individual player [9]. Therefore, detection of the player learning curve appears to be crucial for the realization of an effective gameplay because it reflects player motivation and engagement and can be applied for further adjusting of game difficulty and challenges.

The paper presents an easy and straightforward way of automatic detection of player learning curves as behavioural patterns of player performance, by applying a software component called "Player-centric rule-and-pattern-based adaptation asset" (referred next as *pattern detection asset*) developed in the scope of the RAGE (Realising and Applied Gaming Ecosystem) H2020 project. The component is integrated within a car driving video game in order to detect dynamically specific player learning curves representing overall player performance (OPP) over time. The practical experiments include identification of characteristic patterns in individual learning curves and, next, defining these patterns formally in the asset followed by dynamic pattern detection while playing the game. The found results

provide an answer to the main research question: How the pattern detection asset can be applied in a more complex online 3D video game in order to detect automatically specific learning curves, for further realization of a player-centric adaptation for an improved game playability?

2 RESEARCH METHODOLOGY

The present research aims at automatic detection of behavioral patterns of player performance achieved at runtime of playing a 3D video game for car driving at various weather conditions. The curve of the overall player performance represent the learning curve of an individual player. This section describes the video game, the applied pattern detection asset, and the experimental setup and procedure.

2.1 The car driving video game

We will use a ready cost-free car racing video game to apply the pattern detection asset with research purpose within analyzing and testing benefits of using the proposed game asset. The goal of this game is from one hand to be used like a car simulator and from another hand to be used for improving of the driving skills of players in a more effective, easier and faster way. Therefore, driver and graphical environments of our game have to be as possible as realistic and to represent a realistic representation of a real world scenario (related to simulation). Moreover, it has to be able to detect progress and regression of driving skills of players and to take appropriate action in order to help them improve their achievements (related to learning/improving driving skills).

Taking into account the main goals of the video game we defined the following main requirements it should meet:

- 1 *Attractiveness and realistic* – all racing games are engaging and impressive. Players are able to drive robust luxury cars, to achieve high speed, to be challenged to do complicated maneuvers and solved difficult road situations. Therefore, we need of a 3D video game presenting a realistic road scenarios and creating impressive player experience, motivation and fun.
- 2 *Accessibility* – a video game with online distribution and possibility to be play through a Web browser will allow more participants (players) to take part in our experiment, and thereby, to be received more credible results. In this case our experiment will be independent from time and place.
- 3 *Simplicity* – for a fast start of game playing players need to move quickly into the game and to focus on driving challenges. Consequently, our game needs to have intuitive and user-friendly interface, to support single player mode with simple gameplay without complex rules and to allow a steep learning curve in the beginning adapting the difficulty to the current achievements of a player.
- 4 *Open code* – we need to be able to modify the code of the racing game in order to integrate the pattern detection asset in it, to apply adaptation of game behavior based on it, to logging data for further analysis, etc.

A video game meeting these requirements, appeared to be a 3D car simulator (i.e., a car driving game) having a road passing various terrains and support different road scenarios and challenges. Our choice was determined by the properties of such a gems – because it is attractive, engaging, motivating and makes fun. There are many racing car video games available [10]. However, most of the high quality racing games are not for free or free but not provide sufficiently good power regarding several critical aspects as follows:

- *Realistic and attracting representation* – such as for example: attracting and realistic terrains, smoke of the exhaust tube, drops and vapors when raining, amazing sounds – of the motor, of each collision, of raining, etc.
- *Gauging tools and gadgets* - at least a speedometer, a device showing the current gear, indicator for elapsed and remaining distance, etc.
- *Challenging road scenarios* – terrain of the video game has to provide possibility to players to improve their driving skills and to solve rapidly different road cases. Therefore, in the video game it is needed to be available a road with various types of curves, as well as, it has to be available an environment enabling the availability of different weather conditions such as illumination (day and night), rain, snow, icing and fog.

- *Realistic car mechanic* – for engaging and attracting of players it is required the video game to provide changeable car mechanics and maneuvering such as different levels of friction of tires on the road, sufficient power of the motor and extra keys (e.g., for jumping).

For these reasons, the game was developed by the authors of the paper on the base of an existing video game of car simulator available at GitHub [11]. The existing game was re-designed by adding to it necessary gauging tools, gadgets, and as well as, availability of different weather conditions, and implementation of adaptability based on the pattern detection asset. Hereby, it provides a fascinating first-person car driving simulation with maneuvering on a road with various types of curves, through attracting and realistic terrains. It increases player motivation and fun by visualization of extra tools like a speedometer and trip counter, by playing amazing sounds (of the motor, of each collision, etc.), and by driving under different weather conditions.



Fig. 1. View of the first game prototype of the game.

The learning goal applied in the game is players to develop best skills for driving the car achieving in the same time a maximal velocity and minimum number of collisions at different road scenarios and weather conditions. Under collision it is understood every exit from the road and hitting on the fence guard or on other object. Each collision is characterized by an impact measured on a basis of the current velocity and the angle between the car and the hit object. The higher speed and greater angle lead to greater impact that simulates the degree of the damage in a real-life situation.

The process of adaptation in the game is realized in the following steps:

- 1 *Defining several learning curves* – each one of these learning curves defines a pattern presenting player performance for a period of time. For example, we can define a pattern for increasing/decreasing with 20% of the player performance for a period of 2 minutes. These patterns are defined through the pattern detection asset that supports following features:
 - *Defining of a sequence of absolute or relative values* for time and metric.
 - *Confidence interval of a pattern* - it is presented as percentage of acceptable deviation of a player performance values from the values defined in the pattern.
 - *Confidence level of a pattern* - it is presented as percentage of all player performance values belonging to the values defined in the pattern.

For each one pattern is implemented an event handler that realize a specific game behavior. For example, if a pattern defining increasing of the player performance, the event handler will provoke worsening weather conditions in order to increase the game difficulty and to improve driving skills of the player.

- 2 *Monitoring of the player performance* – the game registers a metric for the overall player performance (OPP) and log its value at every second. It evaluates this metric depending on the current impact of collisions and achieving average speed, according to the following formula:

$$(1) \text{ OPP} = V_{\text{avg}} / (I_{\text{normalized}} + 1),$$

where V_{avg} is the average velocity for a time window of 10 seconds and $I_{normalized}$ is the average normalized impact of collisions calculated following the formula:

$$(2) I_{normalized} = \text{SUM}(V * |\sin(\alpha)|) / N_{coll} / I_{avg},$$

where V is the car velocity at the moment of collision, α is the angle between the car direction and the hit body, N_{coll} is the number of collisions for the time window, and I_{avg} is the average impact value for the game.

- 3 *Detecting matching* between the player curve and one or more of the defined learning curve from Step 1 and *triggering appropriate action* - in case of a found occurrence of specific pattern for the metric or for its feature (such as mean, deviation, or moving average), the component executes an event handler defined for this pattern.

The game story consists of passage of three turns of a terrain containing different type of curves. At each next turn, weather conditions and illumination worsen, making driving increasingly difficult. On the first turn, we have daylight and nice clear weather. On the second lap, the fog goes down and the visibility decreases. On the last turn, it rains, the road becomes slippery and it is getting dark.

We conducted an experiment with this game whose purpose was to be detected type of possible learning curves for each one turn presenting different level of player performance. These learning curves present the evolution of OPP and its mean and moving average values when applying various driving conditions such as fog, rain, and nightfall. The detected player learning curve will be further used for adjusting of the game mechanics (friction to the road at the rain, or/and visibility at the fog, or/and illumination at nightfall) for the individual player, for providing better fun and immersion of the game.

2.2 The game asset for detecting rules and patterns

For automatic detection of player learning curves, the game applies a software component called "Player-centric rule-and-pattern-based adaptation asset" developed in the scope of the RAGE (Realising and Applied Gaming Ecosystem) H2020 project [12]. The component was integrated within the car driving video game in order to detect dynamically specific player learning curves representing OPP over time. The idea of using patterns and rules for adaptation purposes is adopted from the game adaptation control framework using implicit derivation of the player character during the game play developed within the scope of the ADAPTIMITES project [13]. The Rule-and-Pattern-Based Adaptation Asset can be used for two purposes:

- 1 For monitoring of particular player's metrics such as regarding outcomes, emotional status and/or playing style of an individual player:
- 2 For realization of dynamical adaptation according to a specific change of a player metric or its features such as mean, deviation and moving average within a desired time window.

The asset initialization process includes four issues:

- 1 *Registering player-centric metrics* for monitoring whose variation is to be observed by the asset – metrics regarding player's performance (knowledge and intellectual abilities), effective status (emotional experiences, feelings and motivation), playing style, etc..
- 2 Optionally it is allow to be *set a metric feature* for monitoring. Features initially supported are average and moving average for a predefined time window.
- 3 *Setting adaptation triggering rules and patterns* – one or more rules or patterns of change of an already defined metric or its feature are to be defined by using a simple syntax.
- 4 *Definition of adaptation event handlers* – for each adaptation event triggered upon specific rule or pattern, a specific custom event handler will be executed upon the rule/pattern name, acting as adaptation method changing some game features about game mechanics, dynamics and aesthetics [14].
- 5 *Setting the global time and time window at the asset* – the game developer can set/reset the asset timer at any time in order to synchronize it to the game engine.

When the initialization process is complete and the game starts, initially, the game asset receives as input registration requests of player-centric metrics. The metric represent a property of the player such as individual performance (as in the case of this article), emotional status, playing style, etc. Each metric is associated with one or more patterns modeling a game event (e.g. increasing OPP) and

defined by simple formal definitions of rules. The goal of these rules is to be set variation of values of metrics during the play time and they are defined by simple but powerful syntax. Moreover, each pattern can be associated with a feature, such as average or moving average within a desired time window.

After registration of a metric value, the pattern detection asset checks each incoming metric values for occurrence of a rule or a pattern defined for it. The asset looks for absolute or relative changes of a given metric or its feature, and raises a specific event. In case of finding such an occurrence, it fires a triggering event about this rule or pattern and executes its event handler, which has to be defined by the game developer depending on his/her goal to adapt specific game feature(s). Thus, the asset does not depend on any concrete digital game and provides the game developer with freedom to program any control over game adaptation. The software component allows registration of patterns of OPP to be easily defined by the game developer by simple formal definitions including the setting of:

- *Name of the pattern* – it is an identifier of the pattern and it has to be unique.
- *Metric* associated with this pattern – each metric can be associated with one or more patterns
- *Metric feature* – it is optional parameter. The supported features by the pattern detection asset are average and moving average.
- *Set (sequence) of values related to moments of time* – values can be defined as absolute (e.g. 3000, 6000, 9000), relative (e.g. $t+3000$, $t+6000$, $t+9000$) or rule based (e.g. $GT(3000)$ and $LT(9000)$ that means $t > 3000$ and $t < 9000$) moments of time.
- *Set (sequence) of values related to metric value* – as well as the above set of values, they can be defined as absolute, relative and rule based metric values expected for a given moment defined in the set of values related to moments of time.
- *Fitting line* – it is optional parameter and it allows to be defined a range, where metric values may vary. It consists of following two properties:
 - *Confidence interval (accuracy)* – it is acceptable percentage deviation of the registered game metric values from the expected metric value defined initially in the pattern.
 - *Confidence level (recall)* – it is maximum percentage of all registered game metric values that have to be within the defined range (from the confidence interval).

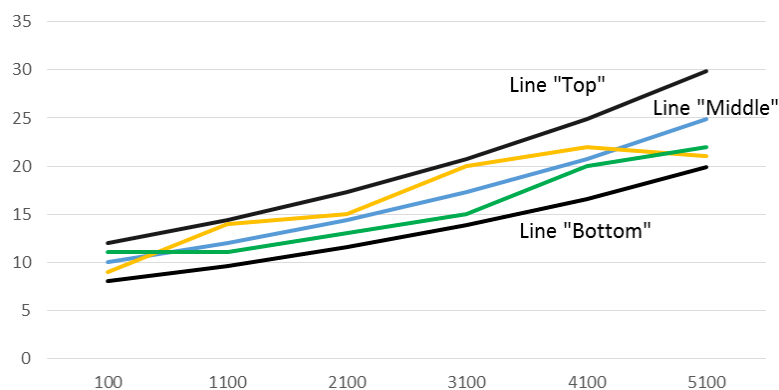


Fig. 2. Pattern example of player performance increasing with 20% each second for a period of 5 seconds.

Examples of simple definitions of patterns and rules are given below:

- Pattern example with absolute feature value using relative time moments after time t namely $t+3000$, $t+6000$ and $t+9000$ milliseconds:

```
{name="GSR mean pattern", metric="GSR", feature="Average", time="t t+3000 t+6000 t+9000",
values="16 20 24 20"}
```
- Pattern example with relative value x of the moving average using absolute time (at the second, fifth and the tenth minute):

```
{name="Happy pattern", metric="happiness", feature="Moving average", time="120000 300000 600000", values="x x+10 x-20"}
```

- Pattern example of player performance that increases with 20% at each second, for a period of 5 seconds, defined with accuracy 20% and confidence level 100%:

```
{name="Increase", metric="Performance", feature="None", time="t t+1000 t+2000 t+3000 t+4000", values="x x*1.2 x*1.2*1.2 x*1.2*1.2*1.2 x*1.2*1.2*1.2*1.2", fitting line={0.8, 1}}.
```

Fig. 2 illustrates this pattern – all curves (formed by sequence of values) that satisfy the pattern are these found between line “Top” and line “Bottom” (in the fig. 2 there are three such lines). The line “Middle” presents the line based on the relative metric value x and relative time moments t defined in the pattern.

2.3 Experimental setup and procedure

For conducting the practical experiment, the game was built for both Windows and browser platforms in order to allow both on-site and online game sessions. Both versions were non-adaptive (i.e., without any player-centric adaptivity), with three levels (each one representing a driving tour) of static difficulty. Each higher level proposed a higher difficulty compared to the previous one. Within a level, the game difficulty was constant. Thus, that the difficulty of the challenges of the game changed in exactly the same way during any playing session for all the players, regardless of a particular player.

For the practical experiment, we invited students at the Faculty of Mathematics and Informatics at Sofia University. FMI-Sofia University, mainly bachelor students of informatics, pedagogy, and software engineering. The participation was entirely anonymous and voluntary. No data about participants were collected. Game sessions were mainly online, at a place and time chosen by the participant, through a Web browser with a Unity plugin. The participants had to follow a very simple procedure: (1) Explanation of the goals, i.e. to drive out three rounds of maintain the highest possible velocity with no or less accidents, passing all the three levels of the game. (2) Game session(s) - playing at least one game session through the three levels of the game, unrestricted in time, with logging the playing time and, as well, the tour and the OPP metric at each second.

The experiments were conducted in two steps:

- Step 1: the participants play the game without using the RAGE asset. The goal was to collect sufficient data about the types of possible learning curves for that game and, next, to find out these types and specific patterns in changes of OPP and its average and moving average.
- Step 2: the same participants play the game again using the same platform; however, the game now applies the RAGE asset, where the specific patterns of changes of OPP are defined in a declarative way, as shown in section 2.2. The goal here was to find out if the asset would detect automatically these patterns during the play sessions and, as well, to verify if the learning curve of given player will be the same as the curve found at Step 1. Expected positive results here will allow us to move to the next stage of experiments addressing adaptation of gameplay according specific patterns in learning curves.

3 RESULTS

The section presents initial results of a longer study of playing the car driving game. In this experiment, nine volunteers have participated by playing the same game twice. The period between the two steps explained over was three days. During that time, we analysed the visual representations of the OPP metric logged in the game once per seconds. As expected, the metric starts from zero and goes consecutively through high and low values by forming local maximums and minimums.

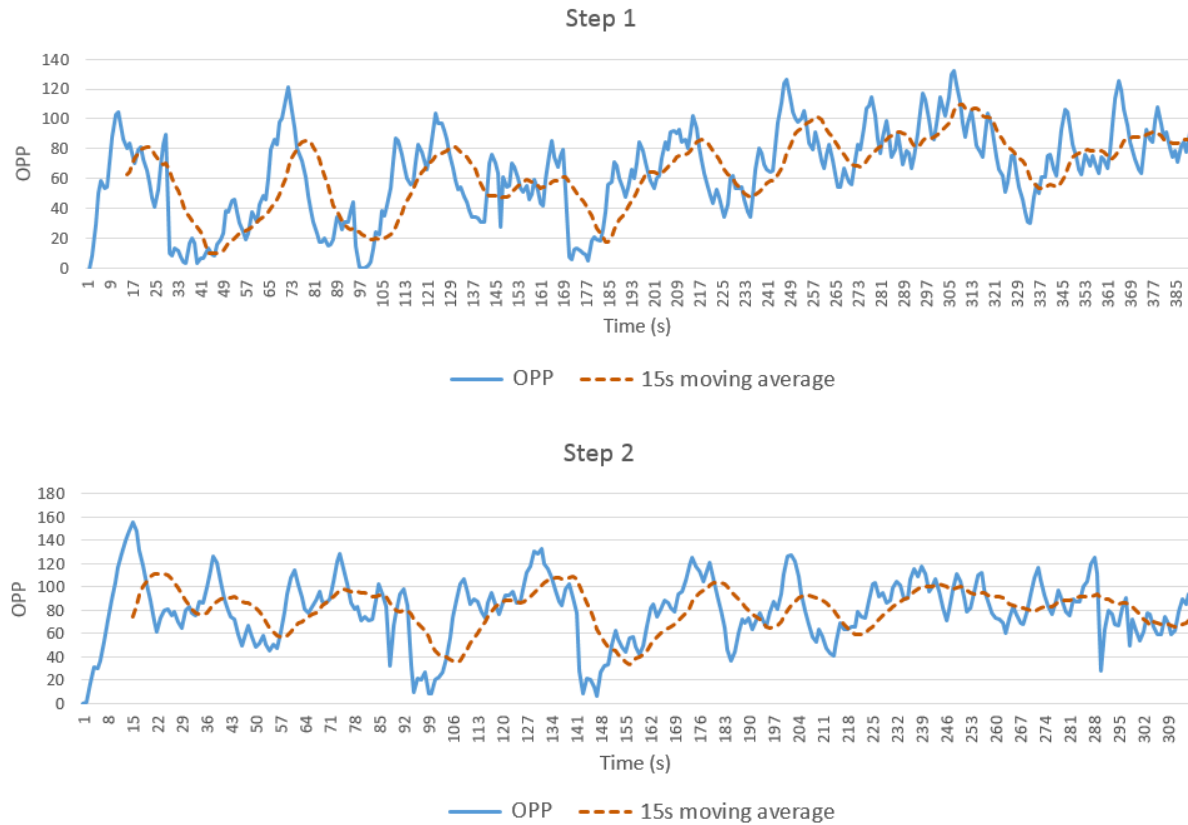


Fig. 3. Charts of OPP and its moving average (within a 15s window) for Step 1 (over) and Step 2 (below) of participant P3.

Fig. 3 represents a typical chart of OPP formed by the metric values obtained from one of the participants during the game sessions until entering the slippery rainy road at Level 2. The first chart provides the results for Step 1, while the second chart shows the OPP achievements for the same participant during Step 2. Besides OPP, the charts present the moving average of this metric calculated within a time window of 15 seconds. We have experimented with shorter and longer time windows but found the value of 15s appears to be most useful because shorter time windows follow the development of the OPP metric, while time windows longer than 15s are too smooth and miss important patterns of OPP. On the other hand, at each second, we have explored the average value of the OPP since the very beginning of the game (not shown in the figure); however, it tends to a constant mean value and therefore cannot reveal what happens with the OPP development.

In both the charts presented in Fig. 3, we can find four specific patterns of increase and decrease of moving average:

- 1 Strong increase – i.e., growing up circa 2.5 times for a period of 20 seconds, e.g. between 57s and 77s, between 113s and 133s, and between 185s and 205s for Step 1;
- 2 Moderate increase – i.e., growing up around 30% for a period of 20 seconds, e.g. between 240s and 260s, and between 340s and 360s for Step 1;
- 3 Strong decrease – i.e., slowing down circa 2.5 times for a period of 20 seconds, e.g. between 20s and 60s, and between 80s and 100s for Step 1;
- 4 Moderate decrease – i.e., slowing down around 30% for a period of 20 seconds, e.g. between 240s and 260s, and between 290s and 310s for Step 2.

These four patterns were found within the learning curves of the other participants, as well. We have coded then inside the RAGE pattern-finder as explained in section 2.2 and have at least two of them for the nine participants during game sessions at Step 2. On the other hand, for both the OPP metric and its 15s moving average, two observations can be drawn:

- 1 The achieved maximal and average OPP for Step 2 appears to be smoother than for Step 1;
- 2 The chart for Step 2 appears to be smoother than the one for Step 1.

The same conclusions can be inferred for the other participants. Initial statistical analysis results are given in Table 1, presenting the mean (average) value of OPP named as OPP_{MEAN} , the standard deviation (OPP_{SD}), the standard error (OPP_{SE}) and, finally, the time for accomplishing the game session. The last column provides the average values of these parameters revealing an increase in OPP_{MEAN} and decrease in OPP_{SD} , OPP_{SE} and the playing time, for steps 1 and 2.

In order to check whether these changes in OPP_{MEAN} , OPP_{SD} , OPP_{SE} , and the playing time measured in Step 1 and in Step 2 are statistically significant, we performed paired two-tiles T-test for the values given in Table 1. Table 2 reveals that the increase in OPP_{MEAN} (at about 19%) and decrease in OPP_{SD} (at about -15%), between Step 1 and Step 2, are statistically significant with $p < 0.05$, while the playing time decreased at circa -17% at value of $p < 0.005$. Therefore, even for nine participant we proved the OPP and its smoothness (indicated by the OPP standard deviation) increases with playing. At the same time, the patterns of playing remain the same – patterns found at Step 1 are detected at Step 2, as well.

Table 1. OPP (mean, standard deviation, and standard error), together with playing time for the nine participants for steps 1 and 2.

Participant		P1	P2	P3	P4	P5	P6	P7	P8	P9	Average
Step 1	OPP_{MEAN}	67.37	68.50	64.17	62.83	59.49	76.90	78.37	72.38	63.84	68.21
	OPP_{SD}	32.46	50.93	29.57	45.62	36.90	34.23	43.19	46.05	39.84	39.87
	OPP_{SE}	1.68	3.00	1.50	2.18	2.04	1.91	3.05	2.84	2.09	2.26
	Time	372	288	389	437	328	321	201	263	362	329
Step 2	OPP_{MEAN}	102.15	57.33	79.90	83.91	96.15	83.24	74.96	79.37	72.10	81.01
	OPP_{SD}	30.51	31.19	28.07	53.75	18.68	25.63	40.98	41.39	34.19	33.82
	OPP_{SE}	1.94	1.96	1.58	3.08	1.16	1.47	3.09	2.58	1.88	2.08
	Time	248	252	314	303	262	305	176	257	331	272

Table 2. Increase and T-test p-value of: mean, standard deviation, and standard error of OPP, and playing time (for steps 1 and 2).

T-test	Increase	p-value
OPP_{MEAN}	18.78%	0.025432
OPP_{SD}	-15.16%	0.043551
OPP_{SE}	-7.68%	0.349085
Time	-17.33%	0.002607

4 CONCLUSIONS

Learning curves (experience curves) remains crucially important for technology enhanced learning and, in particular, for serious games, because they represent how learners (players) do spend time for mastering the learning tasks by acquiring cognitive abilities, knowledge, and necessary skills for solving the challenges of the task (game). Dynamic detection of the learning curve appears to be crucial for any e-learning process, because of several reasons:

- It provides instructors and educators with a picture of the progress in learning and, hence, the performance specific to an individual learner.
- It informs the learner where exactly is he/she in terms of mastering the learning tasks and following the overall learning process;
- It may serve as a means for further adjusting of learning task difficulty, challenges and, as well, of other features of an effective and efficient learning process.

The considerations given above are fully valid for serious video games, as well. The present paper a method for automatic detection of specific learning curves at run time within a 3D video game of car driving in various weather conditions. The method is easy and straightforward thanks to applying a

software component called “Player-centric rule-and-pattern-based adaptation asset” and developed in the scope of the RAGE (Realising and Applied Gaming Ecosystem) H2020 project. This component was integrated within the video game in order to detect dynamically specific player learning curve patterns of overall player performance. The patterns were first identified by logging OPP while playing the game and, next, were defined formally within the asset. We identified these patterns within the learning curves while the same players played the game for a second time, though the mean and the standard deviation of OPP, together with the time for playing the game changed during the second step of the experiment.

Our purpose of runtime detection of learning curve patterns was and remains realization of an efficient and dynamic player-centric adaptation and, hence, improving the overall game playability. Our plan for future works includes dynamic adaptation of various game features, which fall into three main groups [15], namely:

- adaptation of player-driven game tasks and/or game assistance such as helping instructions. As well, the managed appearance of tasks and assistance in their performing during the game flow can be adjusted;
- dynamic adjustment of task difficulty –where the adaptation control is based on the player’s anxiety, or adaptation controlled according the skill level of the player;
- adjustment of properties of audio-visual content and effects, such as ambient light in rooms in a video game.

Thus, our next research is going to address the effect of a dynamic adaptation of the car driving game according to the patterns described here and detected at runtime by the RAGE asset.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the EC H2020 project RAGE (Realising an Applied Gaming Eco-System), Grant Agreement No 644187.

REFERENCES

- [1] S. Egenfeldt-Nielsen, J.H. Smith, S.P. Tosca, “Understanding video games: The essential introduction”. *Routledge*; 2015.
- [2] Boot, A. F. Kramer, D. J. Simons, M. Fabiani, and G. Gratton, “The effects of video game playing on attention, memory, and executive control”. *Acta Psychologica*, 129(3), pp. 387–398, 2008.
- [3] C. S. Green, R. Li and D. Bavelier, “Perceptual learning during action video game playing”. *Topics in Cognitive Science*, 2(2), pp. 202–216, 2010.
- [4] V. J. Shute, S. D’Mello, R. Baker, K. Cho, N. Bosch, J. Ocumpaugh, M. Ventura and V. Almeda, “Modeling how incoming knowledge, persistence, affective states, and in-game progress influence student learning from an educational game”. *Computers & Education*, 86, pp. 224–235, 2015.
- [5] B. Sawyer and D. Rejeski, “Serious games: improving public policy through game-based learning and simulation”. *Woodrow Wilson International Center for Scholars*. Washington, DC, 2002.
- [6] I. Granic, A. Lobel and R.C. Engels, “The benefits of playing video games”. *American Psychologist*, 69(1), p.66, 2014.
- [7] E. Stump. “All about learning curves”. *Society of Cost Estimating and Analysis Conference Paper, Galorath Incorporated., El Segundo, CA*, pp. 1-78, 2002.
- [8] F. Bellotti, R. Berta, A. De Gloria, and L. Primavera. "Adaptive experience engine for serious games", *IEEE Transactions on Computational Intelligence and AI in Games*, 1, no. 4, pp.264-280, 2010.
- [9] C. Linehan, G. Bellord, B. Kirman, Z. H. Morford, and B. Roche. "Learning curves: analysing pace and challenge in four successful puzzle games", *Proc. of the first ACM SIGCHI annual symposium on Computer-human interaction in play*, ACM, pp. 181-190, 2014.

- [10] M. Lebram, H. Engström and H. Gustavsson, "A driving simulator based on video game technology". In *SIGRAD 2006*, (No. 019). Linköping University Electronic Press, 2006.
- [11] Car Simulator Github repository, Open source code, Accessed 18 January, 2018. Retrieved from <https://github.com/rostrage/CarSimulator>.
- [12] W. Van der Vegt, W. Westera, E. Nyamsuren, A. Georgiev, and I.M. Ortiz, "RAGE architecture for reusable serious gaming technology components". *International Journal of Computer Games Technology*, p.3, 2016.
- [13] B. Bontchev, "Modern Trends in the Automatic Generation of Content for Video Games". *Serdica Journal of Computing*, 10(2), pp.133-166, 2017.
- [14] A. Parnandi and R. Gutierrez-Osuna, "A comparative study of game mechanics and control laws for an adaptive physiological game". *Journal on Multimodal User Interfaces*, pp.1-12, 2014.
- [15] B. Bontchev and D. Vassileva. "Affect-based adaptation of an applied video game for educational purposes", *Interactive Technology and Smart Education*, 14.1, pp.31-49, 2017.