DOI: http://dx.doi.org/10.21111/fij.v4i2.3591
Diterima: 27 Oktober 2019 Revisi: 5 November 2019 Terbit: 13 Desember 2019

NPC Braking Decision for Unity Racing Game Starter Kit Using Naïve Bayes

Muhammad Aminul Akbar [1)*], Tri Afirianto [2)], Steven Willy Sanjaya [3)] Ratih Kartika Dewi [4)]

Faculty of Computer Science, Brawijaya University, Indonesia [1,2,3,4)]
muhammad.aminul@ub.ac.id [1)], tri.afirianto@ub.ac.id [2)], steven.willy24@gmail.com [3)], ratihkartikad@ub.ac.id [4)]*

Abstract

Racing video game genre was still being popular today. One way to develop racing games quickly is by using a template or kit that is on the game engine. Racing Game Starter Kit (RGSK) was being the most popular racing game template for Unity game engine. However, there was problem in racing game's NPC especially in RGSK related to NPC vehicle's braking decision. The commonly used method is the Brake Zone, but the developers must manually place the zone themselves in the designated locations for braking. The solution that can be applied for that problem is see the angle formed by the vector of the NPC vehicle with the vector from 2 next following waypoint then determine the best configuration angle threshold for NPC braking, but this also has its shortcoming in which to get the best result, a proper threshold configuration is needed in each track. To resolve the problem, researcher proposed the method of machine learning, Naïve Bayes for the braking decision. Naïve Bayes uses two output classes (brake or no brake) in which the data will be obtained from the player. We use data from players who can control racing car games well or have never hit a wall and have fast lap times. The purpose of this study is to provide an alternative braking method to RGSK that can provide fast lap times but does not affect the game's FPS and without the need to determine or change any parameters on each track. The test result using RGSK v1.1.0a in Unity Game Engine showed that the proposed method can be an alternative method in RGSK braking decisions. Our NPC has faster lap time and was able to prevent the vehicle from crashing with the outer wall without dropping the game's FPS (Frames per Second).

***Keywords**: Braking Decision, Racing Game Starter Kit, Naïve Bayes, Machine Learning, Unity Engine*

Abstrak

Genre video gim balap masih populer saat ini. Salah satu cara untuk mengembangkan game balap dengan cepat adalah menggunakan template atau kit yang ada di game engine. Racing Game Starter Kit (RGSK) adalah templat game balap paling populer pada Unity Game Engine. Namun, terdapat permasalahan NPC pada gim balapan terutama di RGSK terkait dengan keputusan pengereman kendaraan NPC. Metode yang digunakan untuk eksperimen jenis ini adalah Zona Rem. Namun, pengembang harus secara manual menempatkan zona tersebut di lokasi tertentu pada setiap lintasan. Solusi dari masalah ini yang sudah diterapkan pada RGSK v1.1.0a yaitu dapat menggunakan sudut yang dibentuk oleh vektor kendaraan NPC dengan vektor dari 2 titik arah berikutnya, kemudian menentukan ambang sudut terbaik untuk pengereman NPC, tetapi ini juga memiliki masalah yaitu untuk mendapatkan hasil putaran terbaik atau cepat, perlu menentukan konfigurasi ambang batas yang tepat di setiap trek. Untuk mengatasi masalah tersebut, peneliti mengusulkan metode pembelajaran mesin, Naïve Bayes untuk keputusan pengereman. Naïve Bayes menggunakan dua kelas output (rem atau tidak ada mengerem) di mana data akan diperoleh dari pemain. Kami menggunakan data dari pemain yang dapat mengontrol permainan mobil balap dengan baik atau tidak pernah menabrak tembok dan memiliki waktu putaran yang cepat. Tujuan dari penelitian ini adalah untuk memberikan metode pengereman alternatif untuk RGSK yang dapat memberikan waktu putaran yang cepat namun tidak mempengaruhi FPS game dan tanpa perlu menentukan atau mengubah parameter apa pun di setiap trek. Hasil pengujian menggunakan RGSK v1.1.0a di Unity Game Engine menunjukkan bahwa metode yang diusulkan dapat menjadi metode alternatif dalam keputusan pengereman RGSK. NPC kami mempunyai waktu putaran yang lebih cepat dan mampu mencegah kendaraan agar tidak menabrak dinding luar tanpa menjatuhkan FPS game (Frame per Detik).

Kata kunci: *Keputusan Pengereman, Permainan Balap Starter Kit, Naïve Bayes, Pembelajaran Mesin, Unity Engine*

1. INTRODUCTION

Racing game or racing video game is one type of game that has been played since 1969. Racing game is one part of the vehicle simulations genre, which aims to provide an experience of how to drive various vehicles both real and imaginary. The racing game focuses on who gets to the finish line the fastest. Until now, racing games have a large number of enthusiasts, as evidenced by the success of various games such as the Need For Speed series whose entire series in 2009 have sold 100 million copies [1] and sequels are still being made, Need For Speed Payback, which was released November 6, 2017 [2]. There are still many developers who want to develop racing games. One way to develop racing games quickly is using a template or kit that is on the game engine. Racing Game Starter Kit (RGSK) was being the most popular racing game template for Unity game engine [3]. RGSK has been the object of this research, with the aim of helping developers, especially indie developers who use these templates in the unity game engine, the issues raised will be explained in the next paragraph.

The development of racing games is inseparable from the need for NPC (Non-Player Character) which is present in the form of auto-vehicle as the opponent to play against players [4]. NPC is important characters in the game [5]. In the NPC there is a logic given in the form of Pathfinding so that the NPC can drive on the right track [6]. The method that is often used is configuring Waypoints and division of grids with A * [7][8] [9] [10]. The development of physics game is increasingly advanced, causing the vehicle to move to resemble an original vehicle in the real world, so when passing the bend, speed must reduce. One component of NPC's decision making is braking decisions which is about when the NPC should brake. One method that is often used is the brake zone [7], however this method is less effective because each path must be configured manually by installing a brake zone for each bend on all different trajectories. The example use of the brake zone is in the Racing Game Starter Kit (RGSK) v1.0.1 which has now depreciated [11]. Advanced development on RGSK v1.1.0a is the Smart AI System that takes into account the angles formed by the direction vector of the vehicle with the vector of the next waypoint for braking decision[11], however it is not known the most appropriate threshold angle to obtain the best braking performance in each track. In this study, we raise the issue of braking decisions at RGSK latest version v1.1.0a. The braking method in RGSK v1.1.0a is the same as the method in [9] and [8] research.

Several previous literatures regarding the development of NPC's behavior in racing games [9], [12], and [13]. In a recent work ([9] and [12]) used TORCS The Open Racing Car Simulator. In [9], TORCS was used to make simulated car racing championship and used angle and track sensors to determine the condition of the road (the road is straight or turn). However the [9] scenario different with RGSK, sensor in RGSK is used to knowing other vehicle around the NPC itself. In [12] using genetic algorithm and Bezier curves to get optimal racing line. Racing line in [12] is same as waypoints in RGSK that is the line to follow to achieve the best lap-time possible on a given track, however braking decisions are still needed so that NPC vehicles do not leave the racing line. [13] compares the neural network and behavior base approach in making vehicle control simulations in racing games. Both methods succeed in defeating 3 standard heuristic controllers. However [13] have different basic system with RGSK especially in waypoints. In [13] the scenario is two waypoints are visible on the competition field unlike in RGSK, because this template is used to develop a racing game so we can access all the waypoints, and we should be able to take advantage of that. The approach from [13] which was use machine learning to develop the vehicle controller then we adapted in this research by improvising the input and use of real human players to get sample data. Another previous research [14] is using unity game engine to build racing game, however [14] research about improving the balance of players in racing games by adjusting several variables such as the speed of the player in order to keep up with other player.

We proposed naïve bayes machine learning to solve these problems and used three inputs there were speed, turn's angle, and distance between character and next turn. Sample data is obtained from human players who can play the RGSK standard game well without going off the track and having a fast time. The purpose of using a human player is for the NPC to learn the behavior of the player. Naïve Bayes is a probability-based classification algorithm that uses the Bayes theorem with an assumption that features that describe objects are not statistically bound to one another or independent [15]. Assumption owned by Naïve Bayes is rarely fulfilled in conditions in the real world, even so the Naïve Bayes approach can still function well even though its assumptions are not fulfilled. The training process and tests on Naïve Bayes can also be done quickly and are more tolerant of missing data compared to the Bayes Network classification. The Naïve Bayes method has been widely applied in various fields, several fields that are often used and proven to have effective performance, there are Real time Prediction, Text Classification, and Recommendation System. Naïve Bayes is also often used to help diagnose diseases, for example in heart disease [16]. Text classification has also been used mainly in spam filtering and also used in song lyrics [17]. Naïve Bayes has also begun to be examined in the field of games for example in the prediction of the RTS Game strategy [18] with 83.7% precision and 76.7% recall, predicted results in DOTA 2 [15] with accuracy 85.33% and for opening prediction in RTS Starcraft Game[19], NBA Game Result Prediction with 80% accuracy [20] . Naïve Bayes has a fast calculation speed and does not consume large computing resources [21]. So that it is

expected that Naïve Bayes will be effective to become an NPC braking decision that can be applied to racing games with accurate and efficient performance and better than the Smart AI System on RGSK.

2. RESEARCH METHOD

This study aims to create alternative method for braking decisions system at RGSK latest version v1.1.0a using naïve bayes method, problems from existing braking decision system has been explained in introduction. The steps in this study are determining the features or inputs for naive bayes system, designing braking decision systems, Implementation in RGSK unity game engine, data collection and testing.

2.1. INPUTS FOR NAIVE BAYES SYSTEM

We have decided to use 3 input features and two output classes for naïve bayes system. Those three input features are speed, turn's angle, and distance between character and next turn. Those three features were inspired by H. Tang research [13] that use speed, distance and vector for his autonomous vehicle rule base. Two output classes for naïve bayes system are brake and no brake.

The first feature is speed, that monitored by currentSpeed variable in default RGSK script. This feature is very influential on the decision to take the brake, because in low speeds the use of brakes is less frequent even though it will pass a sharp bend. At high speeds the brakes are more needed, especially when going through corners. Second and third features are turn's angle and turn's distance as illustrated in Figure 1. Waypoints is the method for NPC navigation in the RGSK, therefore there are many waypoints placed along the circuit as seen in Figure 2 in the form of a green circle. The second and third features are calculated using vectors formed from waypoints along the circuit.

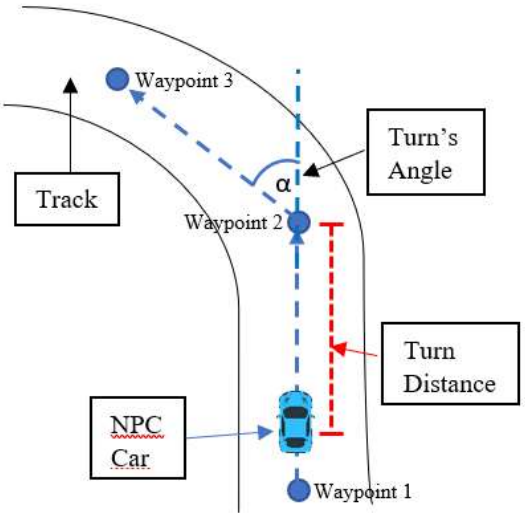

Figure 1. *Turn's distance and angle*

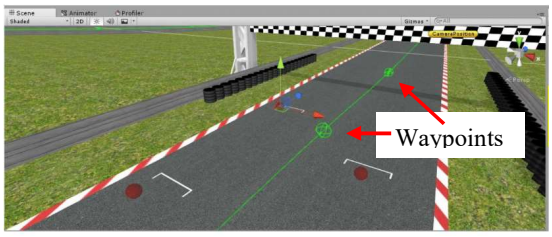

Figure 2. *RGSK Waypoints.*

2.2. DESIGN AND IMPLEMENTATION BRAKING DECISION SYSTEM.

Braking decision system is designed by applying naïve bayes algorithm. The output of naïve bayes is a braking decision whether to brake or not. Naïve bayes is a classifier known to be simple and very efficient. It is called bayes because the probabilistic model is based on Bayes's theorem, and naive adjectives come from the assumption that features in a dataset are mutually independent. In practice, the independence assumption is often violated, but naive bayes classifiers still tend to perform very well under this unrealistic assumption, especially for small sample sizes, naive bayes classifiers can outperform the more powerful alternatives[22]. Naïve bayes equation is:

$$p(C_k|x) = \frac{p(C_k)p(x|C_k)}{p(x)} \quad (1)$$

Where $p(C_k|x)$ is posterior probability, $p(C_k)$ is prior probability, $p(x|C_k)$ is likelihood and $p(x)$ is evidence. Naïve bayes for continuous data using Gaussian Probability which the equation is:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (2)$$

We also use normalization for the system, the normalization used in the system is min-max normalization, which has the following equation:

$$X^* = \frac{X - \min(X)}{max(X) - \min(X)} \quad (3)$$

All those three-equation used inside the Naïve bayes system, first the training data normalized with min-max normalization (3), then enter the training phase for getting average, variance, and prior probability. In the testing phase, test data has also been normalized then calculated inside the naïve bayes for getting the braking decision between 1 (braking) and 0 (not braking). Figure 3 and 4 show our naïve bayes system flowchart. Inputs for naïve bayes consisting of speed, turn's angle, and distance between character and next waypoint are taken in a fixed update Unity life cycle. We added a new function to the OpponentController script in RGSK, we named it bayesBrake. bayesBrake is a function that contains naïve bayes algorithm that shown in Figure 3 and 4. OpponentController is a script in

RGSK that is used to control NPC vehicles in RGSK. OpponentControler script contains navigateAi function. navigateAI is the main component to run the steering and throttling process on the NPC. We modified the navigateAi function by adding a process called the bayesBrake function. Figure 6 shows the distinction of program flow that occur in navigateAi before and after modification. We add the bayesBrake function call before the feedInput function call. By default RGSK has 2 kinds of methods to handle braking, first using the brake zone placed on the OnTrigger lifecycle, this is in the initial version of the RGSK and the second uses the maximum threshold angle that the NPC must consider braking, this is the last version of the RGSK. In this study RGSK has used the second braking method that is processed in the Acceleration, Braking & Steering Calculation section in the navigateAI function. We modified the workflow in navigateAI by eliminating the braking calculation process and replacing it by adding the bayesBrake function call before the feedInput function call. The feedInput function accepts the results from the previous calculation process in the form of values for braking, steering and acceleration and continues to the process of moving the vehicle according to the value received. The pseudo code for navigateAI can be shown in Figure 5.

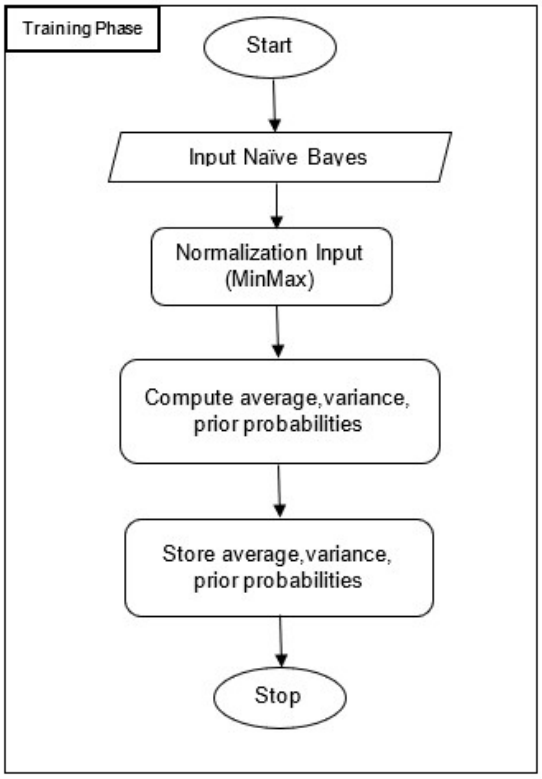

Figure 3. *Naïve Bayes Training Phase.*

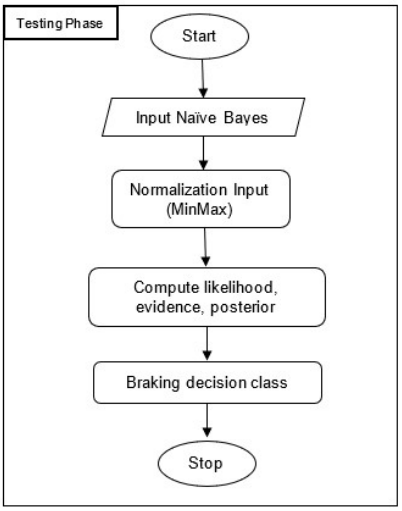

Figure 4. *Naïve Bayes Testing Phase.*

```
1  void NavigateAI()
2  {
3    Calculate AccelerationNSteering()
4    // code we add is in in below
5   float mybrake
6   Transform              w1              =
7  stats.lastPassedNode.position
8    Transform              w2              =
9  stats.NextlastPassedNode.position
10   Transform              w3              =
11 stats.TwoAheadNode.position
12   float distance = Vector3.Distance(w2,
13 this.transform.position)
14   var w3w2 = (w3 - w2).normalized
15   var w2w1 = (w2 - w1).normalized
16   float              angle               =
17 Vector3.Angle(w2w1,w3w2)
18   mybrake                                =
19 bayesBrake.testing(currentSpeed, angle,
20 distance)
21
22 //Finaly, feed the input values ,
23 //throttle and steer are obtained from
24 //the AccelerationNSteering calculation
25 //process
26
27   FeedInput(throttle, mybrake, Steer)
28 }
```

Figure 5. *navigateAI code*

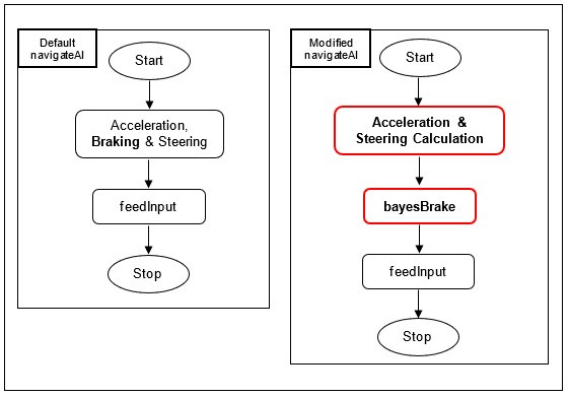

Figure 6. *navigateAI function*

2.3. DATA COLLECTION

There are three input features and two output classes used in the design. Those three features are speed, turn's angle, and distance between character and next turn as illustrated in Figure 1. We gather the data from human player which has lap's time faster than default RGSK Hard Bot and never both go off-road and hit the wall. Data captured every 0.02 second and every time brake was used. Player decision recorded with the value 1 for braking and 0 for not braking. Some sample data from player can be seen in Table 1.

Table 1. *Sample data from player*

No	Speed (mil/h)	Turn angle (degree)	Distance	Brake
1	0	1.206259	26.7355	0
2	58	0.5967609	8.97639	0
3	108	4.589408	42.2416	0
4	121	21.60911	7.30295	1
5	116	14.32146	9.96055	1
6	138	19.42491	20.2914	0
7	140	19.42491	4.551	1
8	90	33.28666	32.8786	1
9	74	35.34315	31.0063	0
10	150	16.94625	14.55	0

2.4. TESTING SCENARIO

Accuracy testing and three performance testing are used in this research. The simplest approach to measuring accuracy is to separate randomly available data into a set of training to produce a learning model and a set of tests to measure accuracy. This method, sometimes called a cross-validation holdout. However, this has a disadvantage that is if we use half the data for the test set, then we only train half the data, and we might get a bad hypothesis [23]. To be able to use lots of sample data and get accurate estimates, we used a technique called k-fold cross validation. In the k-fold cross validation, each sample data has a dual role as training data and test data. First, we divide the data into sets of equal parts. then do a learning round. At each round, 1 / k sample data is held as a test set and the remaining sample data is used as training data [23].

The other test is frame per second (fps) testing, lap time and off-road testing. We test the FPS to find out if there are differences in the FPS between the method we propose and the default method of the RGSK. frames per second is the most commonly used metric as a performance evaluation of video games. A rule of thumb that most games are very fun to play with frame rates above 30 FPS. But games currently trying to reach a frame rate of 60 FPS in order to synchronize with the current screen [24]. The proposed Naive Bayes NPC and the Hard RGSK NPC goes through the circuit for 10 rounds on its own, then we capture the FPS using unity game engine tool.

For lap time testing, Naïve Bayes NPC races against RGSK Hard Bot on the circuit for 10 laps. We chose RGSK Hard Bot because it is the default NPC RGSK that has the best racing ability. We want to know the lap time performance of the proposed method when compared to the best default NPC from RGSK. For offroad testing, the proposed Naive Bayes NPC and the Hard RGSK NPC goes through the circuit for 10 rounds on its own, then we analyzed the path taken by the two NPCs.

3. RESULTS AND DISCUSSION

K-Fold Cross validation testing and three performance testing are used in this research, Detailed explanation about each result and analysis is presented in the next few sections. We compared our proposed method with default NPC that is available in RGSK v1.1.0a. We chose the hardest NPC level because the NPC has the fastest lap time on the RGSK template. The braking method in RGSK v1.1.0a is the same as the method in Loiacono's research [13] and chan [8] which use angle value to determine braking decision. Brake Zone method was depreciated in RGSK template, so we didn't compare with it.

3.1. K-FOLD CROSS VALIDATION

This test is done to get the accuracy and best classification model. This test is Table 2 shows the result of 5-Fold Cross validation, total 1958 samples data separated 80:20 for training set and testing set. Samples data rotated five times for cross validation to get the accuracy percentage. The average accuracy of the system as showed in Table 2 is 83.21%**.** The best classification model is rotation number 5. Then we use model 5 in another test.

Table 2*. Accuracy Testing*

Rotation (k)	Correct	Wrong	Percentage (%)
1	299	93	76.2
2	368	24	93.8
3	340	52	86.7
4	232	160	59.1
5	392	0	100
Average	326.2	65.8	83.21

3.2. FPS TEST

We test the FPS to find out if there are differences in the FPS between the method we propose and the default method of the RGSK. Table 3 shows the average FPS for both NPC for every lap in 10 laps. Both NPC have stable 60 FPS for every lap with only 0.013 frame average difference. We conclude that there is no difference in fps or decrease in fps when using the proposed method. This game should still be smoothly run or can get good fps when played with a system that meets the minimum requirements.

Table 3. *FPS Average*

Lap	FPS (Bayes)	FPS (NPC Hard)
1	60.38596	60.3853
2	60.38918	60.36808
3	60.38906	60.36131
4	60.36587	60.3387
5	60.39067	60.35949
6	60.35355	60.34107
7	60.38358	60.39008
8	60.3833	60.35792
9	60.38409	60.38578
10	60.36745	60.37215
Average	60.379271	60.365988

3.3. Lap Time Test

This test is done to know the lap time performance of the proposed method when compared to the best default NPC from RGSK that is Hard NPC. Table 4 shows the result of the race for 10 laps. Both of NPC has stable time in lap 2 to lap 10. The difference in lap times in the first round is due to the two NPCs starting with speed = 0 at the start line and starting to accelerate to maximum speed, but for the rest of the rounds, the two NPCs are at their optimal speed when they cross the start or finish line. Naïve Bayes NPC wins over the Hard NPC for every lap through 10 laps. With total time difference 54.80 seconds and average time difference 5.48 seconds. As the race proceed, the distance between two NPC getting further away as shows in Figure 7 and with enough lap, Naïve Bayes NPC could overlap Hard NPC. We still have to look at the next test, which is the offroad test, because even though the NPC Naive Bayes is faster, there is still a possibility that this is due to the path taken by the Naive Bayes NPC out of the asphalt track since poor braking decisions.

Table 4. *NPC Lap Time*

Table Lap	Bayes(mm:ss)	Hard(mm:ss)
1	01:01,86	01:05,63
2	00:51,62	00:57,25
3	00:51,42	00:57,40
4	00:51,60	00:57,03
5	00:51,81	00:56,93
6	00:51,52	00:57,11
7	00:51,76	00:57,14
8	00:51,74	00:57,15
9	00:51,22	00:57,29
10	00:51,54	00:57,96
Total	08:46,09	09:40,89
Average	00:52,61	00:58,09

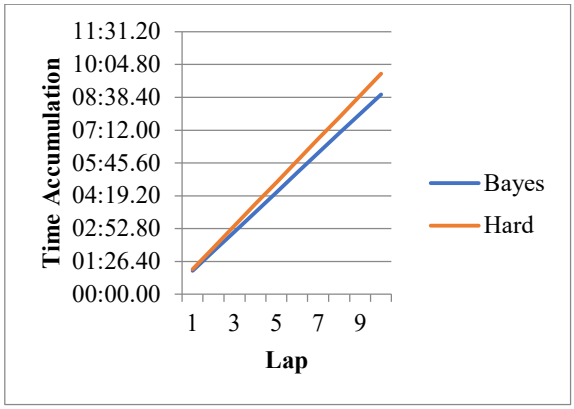

Figure 7. *Time Accumulation*

3.4. Offroad test

For the off-road testing, both Naïve Bayes NPC and Hard RGSK NPC race alone for 10 laps, then we have drawn the path taken by the Naïve Bayes NPC with yellow line (Figure 8) and the Hard RGSK NPC with green line (Figure 9). Naïve bayes NPC hit the grass outside the road (offroad) in two area indicated by red line that can be shown in Figure 8, but never hit the boundary wall outside the road. While hard NPC as shows in Figure 9, only hit one area the same as the first area in Figure 8. All hit areas occur on a straight track after the vehicle turns at the corner. We have analyzed why Naive Bayes NPCs are more frequently hit the grass, this is due to the turning angle that didn't consider the vehicle direction, in some case even in the straight track, the NPC's vehicle could have some angle that formed by the direction of the vehicle with the next waypoint vector after turning in previous corner, if the turn's angle didn't count the vehicle direction in that case (straight track after the vehicle turns at the corner), naïve bayes NPC thinks that it's a safe angle (because straight track) and didn't need to use brake.

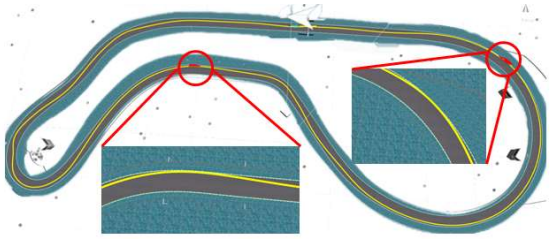

Figure 8. *Naïve Bayes NPC Track line*

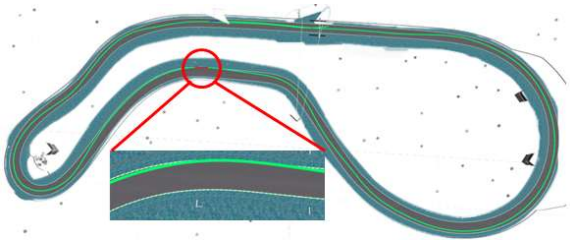

Figure 9. *Hard NPC Track line*

4. CONCLUSION

The proposed method has been applied to Naive Bayes NPC and as we expected, the method we propose can be an alternative method for NPC braking decisions on RGSK. As seen in the test results, the average accuracy is quite good, 83.21%. Naive Bayes NPCs can provide faster lap times than the default Hard NPC, keeping the vehicle from crashing into the walls and having no FPS difference or FPS reduction when compared to the default RGSK method. However there still needs to be improvement, Naive Bayes NPCs still hit the grass more than NPC Hard. For future research, improvements can be made by considering vehicle direction as seen in off-road test the NPC's vehicle could have some angle that formed by the direction of the vehicle with the next waypoint vector after turning in previous corner, if the turn's angle didn't count the vehicle direction in that case (straight track after the vehicle turns at the corner), naïve bayes NPC thinks that it's a safe angle (because straight track) and didn't need to use brake.

5. REFERENCES

[1] F. Totu, "100 million Need for Speed Games Have Been Sold to This Day." .

[2] Electronics Arts, "Need For Speed Payback.," 2017. .

[3] Unity, "Unity Asset Store." .

[4] J. Wang, "Classification of Humans and Bots in Two Typical Two-player Computer Games," in *2018 3rd International Conference on Computer and Communication Systems (ICCCS)*, 2018, pp. 502–505.

[5] I. Mabruroh and D. Herumurti, "Adaptive Non Playable Character in RPG Game Using Logarithmic Learning For Generalized Classifier Neural Network (L-GCNN)," *Kinet. Game Technol. Inf. Syst. Comput. Network, Comput. Electron. Control*, vol. 4, no. 2, p. 127, 2019.

[6] Y. Sazaki, A. Primanita, and M. Syahroyni, "Pathfinding car racing game using dynamic pathfinding algorithm and algorithm A∗," in *Proceedings - ICWT 2017: 3rd International Conference on Wireless and Telematics 2017*, 2018, vol. 2017-July, pp. 164–169.

[7] C. Bennett and D. V. Sagmiller, *Unity AI Programming Essentials*. Packt Publishing Limited, 2014.

[8] M. T. Chan, C. W. Chan, and C. Gelowitz, "Development of a Car Racing Simulator Game Using Artificial Intelligence Techniques," *Int. J. Comput. Games Technol.*, vol. 2015, pp. 1–6, 2015.

[9] D. Loiacono *et al.*, "The 2009 simulated car racing championship," *IEEE Trans. Comput. Intell. AI Games*, vol. 2, no. 2, pp. 131–147, 2010.

[10] Y. Sazaki, H. Satria, and M. Syahroyni, "Comparison of A∗ and dynamic pathfinding algorithm with dynamic pathfinding algorithm for NPC on car racing game," *Proceeding 2017 11th Int. Conf. Telecommun. Syst. Serv. Appl. TSSA 2017*, vol. 2018-Janua, pp. 1–6, 2018.

[11] I. Game, "Racing Game Starter Kit," 2016. .

[12] M. Botta, V. Gautieri, D. Loiacono, and P. L. Lanzi, "Evolving the optimal racing line in a high-end racing game," *2012 IEEE Conf. Comput. Intell. Games, CIG 2012*, pp. 108–115, 2012.

[13] H. Tang, C. H. Tan, K. C. Tan, and A. Tay, "Neural network versus behavior based approach in simulated car racing game," *2009 IEEE Work. Evol. Self-Developing Intell. Syst. ESDIS 2009 - Proc.*, vol. 117576, pp. 58–65, 2009.

[14] J. E. Cechanowicz, C. Gutwin, S. Bateman, R. Mandryk, and I. Stavness, "Improving player balancing in racing games," *Proc. first ACM SIGCHI Annu. Symp. Comput. Interact. Play - CHI Play '14*, pp. 47–56, 2014.

[15] K. Wang and W. Shang, "Outcome prediction of DOTA2 based on Naïve Bayes classifier," *Proc. - 16th IEEE/ACIS Int. Conf. Comput. Inf. Sci. ICIS 2017*, no. 1994, pp. 591–593, 2017.

[16] M. A. Jabbar and S. Samreen, "Heart disease prediction system based on hidden naïve bayes classifier," *2016 Int. Conf. Circuits, Control. Commun. Comput. I4C 2016*, pp. 1–5, 2017.

[17] Y. An, S. Sun, and S. Wang, "Naive Bayes classifiers for music emotion classification based on lyrics," *Proc. - 16th IEEE/ACIS Int. Conf. Comput. Inf. Sci. ICIS 2017*, no. 1, pp. 635–638, 2017.

[18] W. Hamilton and M. O. Shafiq, "Opponent resource prediction in starcraft using imperfect information," *Proc. - 9th IEEE Int. Conf. Big Knowledge, ICBK 2018*, pp. 368–375, 2018.

[19] G. Synnaeve and P. Bessière, "A Bayesian model for opening prediction in RTS games with application to StarCraft," *2011 IEEE Conf. Comput. Intell. Games, CIG 2011*, pp. 281–288, 2011.

[20] F. Thabtah, L. Zhang, and N. Abdelhamid, "NBA Game Result Prediction Using Feature Analysis and Machine Learning," *Ann. Data Sci.*, vol. 6, no. 1, pp. 103–116, 2019.

[21] A. Ashari, I. Paryudi, and A. Min, "Performance Comparison between Naïve Bayes, Decision Tree and k-Nearest Neighbor in Searching Alternative Design in an Energy Simulation Tool," *Int. J. Adv. Comput. Sci. Appl.*, vol. 4, no. 11, pp. 33–39, 2013.

[22] S. Raschka, "Naive Bayes and Text Classification I - Introduction and Theory," pp. 1–20, 2014.

[23] S. J. Russell and P. Norvig, *Artificial Intelligence A Modern Approach;*

PearsonEducation. 2003.

[24] K. T. Claypool and M. Claypool, "On frame rate and player performance in first person shooter games," *Multimed. Syst.*, vol. 13, no. 1, pp. 3–17, 2007.