DOI: http://dx.doi.org/10.21111/fij.v7i1.5487
Diterima: 12 Januari 2021 Revisi: 30 April 2021 Terbit: 28 Oktober 2021

# Pengembangan *Non-Player Character (NPC)* Menggunakan Unity ML-Agents Pada Karting Microgame

**Muhammad Yasir Anshari Haq [1)*], Muhammad Aminul Akbar [2)], Tri Afirianto [3)]**

*Universitas Brawijaya [1,2,3)]*
yasir.haq98@gmail.com [1)*], muhammad.aminul@ub.ac.id [2)] tri.afirianto@ub.ac.id [3)]

## Abstrak

*Perkembangan teknologi di bidang gim sekarang sudah sangat pesat terutama pada gim balapan. Gim balapan memiliki tujuan untuk memberikan pemain sebuah pengalaman yang menantang dan menyenangkan dalam sebuah balapan melawan mobil yang dikendalikan oleh gim tersebut atau biasa disebut dengan Non-Player Character (NPC). Pengembangan gim balapan tentunya tidak dapat lepas dari pengembangan NPC sebagai lawan main dari pemain. Pada umumnya NPC dikembangkan menggunakan metode waypoint untuk navigasi dalam melintasi trek balapan. Kekurangan dalam metode waypoint adalah harus diatur secara manual untuk setiap trek dan memakan waktu yang lama untuk mengatur waypoint pada setiap trek. Begitu juga untuk membuat NPC balap yang kompetitif dibutuhkan desain rule base yang kompleks. Peneliti mengusulkan menggunakan metode machine learning untuk mengatasi permasalahan tersebut. Unity3D menyediakan sebuah open-source API bernama Unity ML-Agents yang dapat digunakan untuk melatih NPC. NPC dilatih menggunakan metode reinforcement learning dengan Unity ML-Agents yang bertujuan untuk melatih NPC dengan cara memberikan reward agar mencapai hasil yang optimal. Hasil yang didapatkan dengan memanfaatkan Unity ML-Agents adalah NPC yang dapat melintasi berbagai macam trek dan dapat menghindari tabrakan. NPC yang telah dikembangkan dengan Unity ML-Agents juga mendapatkan waktu total yang lebih sedikit dibandingkan dengan waktu total yang diperlukan pemain untuk menempuh 3 lap putaran pada suatu trek yaitu 55,9 detik dibandingkan dengan 59,4 detik.*

**Kata kunci:** *gim balapan, non-player character, unity ml-agents, reinforcement learning.*

## Abstract

***[Non-Player Character (NPC) Development Using Unity ML-Agents in Karting Microgame]*** *Nowadays, the development of game technology is very fast, especially in racing games. The racing game aims to provide players with a challenging and fun experience in a race against cars controlled by the game or commonly known as the Non-Player Character (NPC). Of course, the development of a racing game cannot be separated from the development of NPCs as opponents of the players. In general, NPCs were developed using the waypoint method for navigation across racetrack. The disadvantage of the waypoint method is that it must be set manually for each track, and it takes a long time to set the waypoint for each track. Likewise, making a competitive racing NPC requires a complex rule base design. Researchers suggest using machine learning methods to overcome these problems. Unity3D provides an open-source API called Unity ML-Agents which can be used to train NPCs. NPCs are trained using the reinforcement learning method with Unity ML-Agents which aims to train NPCs by providing rewards to achieve optimal results. The results obtained by utilizing Unity ML-Agents are NPCs that can traverse various kinds of tracks and can avoid collisions. NPCs that have been developed with Unity ML-Agents also get less total time compared to the total time required for a player to take 3 laps on a track, which is 55.9 seconds compared to 59.4 seconds.*

**Keywords:** *racing game, non-player character, unity ml-agents, reinforcement learning.*

## 1. PENDAHULUAN

Perkembangan teknologi di bidang gim sekarang sudah sangat pesat terutama pada gim balapan. Gim balapan merupakan salah satu jenis gim yang sudah ada sejak tahun 1974 [1]. Gim balapan memiliki tujuan untuk memberikan pemain sebuah pengalaman yang menantang dan menyenangkan dalam sebuah balapan melawan mobil yang dikendalikan oleh gim tersebut atau biasa disebut dengan NPC [2]. Gim balapan merupakan gim yang populer dan memiliki banyak peminat, hal ini dibuktikan dengan suksesnya beberapa gim balapan seperti Asphalt 8: Airbone yang sudah diunduh lebih dari 100 juta pengguna pada Google Playstore. Seri

terbaru dari gim Asphalt adalah Asphalt 9: Legends yang dirilis pada tanggal 25 Juli 2018 [3] dan sudah diunduh lebih dari 10 juta pengguna pada Google Playstore. Gim balapan lain yang juga sukses adalah seri Mario Kart, salah satunya adalah Mario Kart 8 Deluxe yang menjadi penjualan terbanyak dengan total penjualan 22 juta kopi pada *platform Nintendo Switch* [4]. Seri terbaru dari Mario Kart adalah Mario Kart Tour yang rilis pada tanggal 25 September 2019 yang sudah diunduh lebih dari 50 juta pengguna pada Google Playstore.

Pengembangan gim balapan dapat dilakukan dengan menggunakan *game engine* Unity3D. Unity3D banyak digunakan dalam industri gim untuk mengembangkan gim. Hal ini dikarenakan Unity3D mendukung pembuatan gim di berbagai macam *platform* [5]. Untuk memudahkan pengembangan gim balapan, tersedia *game template* yang bernama Karting Microgame pada Unity3D *Asset Store*. Karting Micrograme membebaskan pengguna untuk melakukan modifikasi pada gim sesuai dengan keinginan pengguna. Hal ini bertujuan untuk memudahkan pengguna mempelajari dasar dari Unity3D [6]. Karting Micrograme sudah dilengkapi dengan fitur seperti *Kart Movement, Keyboard Input, Track Manager, Checkpoint, Race Countdown, Timer,* dan *Lap*, tetapi pada *game template* tersebut belum terdapat *Non-Player Character* (NPC) sebagai lawan main dari pemain.

Tanpa adanya NPC sebagai lawan main dari pemain tentunya tujuan gim balapan tidak dapat terpenuhi. Maka dari itu, dalam pengembangan gim balapan tidak dapat terlepas dari pengembangan NPC sebagai lawan main dari pemain. Secara umum, NPC pada gim balapan dikembangkan menggunakan metode *waypoint* untuk navigasi dalam melintasi trek balapan, tetapi metode *waypoint* memiliki kelemahan utama yaitu *waypoint* harus diatur secara manual untuk setiap trek dan memakan waktu yang lama untuk mengatur *waypoint* [7]. Begitu juga untuk membuat npc balap yang kompetitif dibutuhkan desain rule base yang kompleks. Peneliti mengusulkan menggunakan metode *machine learning* untuk mengatasi permasalahan tersebut. Metode *machine learning* memungkinkan NPC untuk meningkatkan performa dengan cara belajar dari kesalahan dan keberhasilan atau dengan meniru taktik dari pemain [8]. Unity3D menyediakan *open-source* API bernama Unity ML-Agents yang dapat digunakan untuk menyimulasikan *learning environment* dalam melatih NPC [9]. Salah satu metode yang dapat digunakan untuk melatih NPC dengan Unity ML-Agents adalah *reinforcement learning* yang bertujuan melatih NPC dengan memberikan *reward* agar mencapai hasil yang optimal [10]. Kelebihan dalam menggunakan *reinforcement learning* adalah NPC melakukan pembelajaran dengan sendirinya melalui proses *trial and error*. Oleh karena itu, *reinforcement learning* tidak memerlukan data set besar untuk mulai melatih NPC [11].

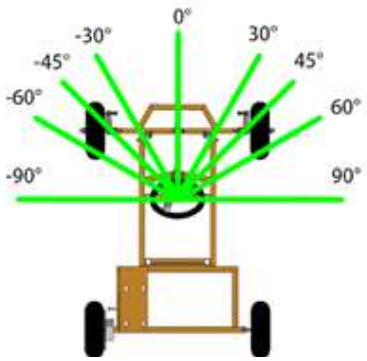
Dalam penelitian ini, peneliti ingin mengembangkan NPC pada *game template* Karting Microgame. Metode yang digunakan adalah *reinforcement learning* untuk melatih NPC dengan menggunakan API Unity ML-Agents Toolkit untuk menyimulasikan *learning environment*. Terdapat beberapa penelitian terdahulu dengan topik penelitian serupa yang telah dilakukan. Salah satunya adalah penelitian yang telah dilakukan oleh Glavin dan Madden [12] yang menghasilkan NPC yang telah dikembangkan menggunakan *reinforcement learning* mendapatkan *kill streak* yang lebih tinggi, *kill-death* rasio yang lebih tinggi dan akurasi menembak yang lebih tinggi dibandingkan dengan NPC yang tidak menerapkan *reinforcement learning*. Selain itu, terdapat penelitian lainnya yang telah dilakukan oleh Wender dan Watson [13] dengan hasil NPC yang telah dikembangkan menggunakan *reinforcement learning* dapat mengalahkan NPC bawaan pada gim *Real-Time Strategy* StarCraft:Broodwar dengan tingkat kemenangan mencapai 100%. Maka dari itu, diharapkan dengan menggunakan *reinforcement learning* NPC yang dikembangkan pada Karting Mikrogame menjadi NPC yang dapat melintasi trek dan menghindari tabrakan.

## 2. BAHAN DAN METODE

Penelitian ini bertujuan untuk membuat *non-player character* (NPC) dengan metode *reinforcement learning* menggunakan API Unity ML-Agents pada *game template* Karting Microgame. NPC dilatih agar bisa melakukan akselerasi dan *steering* untuk melintasi trek dan menghindari tabrakan. Langkah-langkah yang dilakukan pada penelitian ini adalah perancangan, implementasi, dan pengujian.

### 2.1. Perancangan

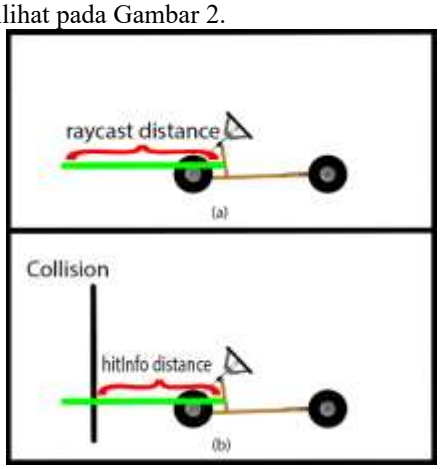
Pada tahap ini dilakukan perancangan, yaitu perancangan arsitektur NPC yang dibagi menjadi empat tahapan, yang meliputi perancangan sensor *raycast*, perancangan skenario, perancangan diagram alir dan perancangan *class diagram*. Perancangan sensor raycast dapat dilihat pada Gambar 1 dan Gambar 2.




**Gambar** Error! No text of specified style in

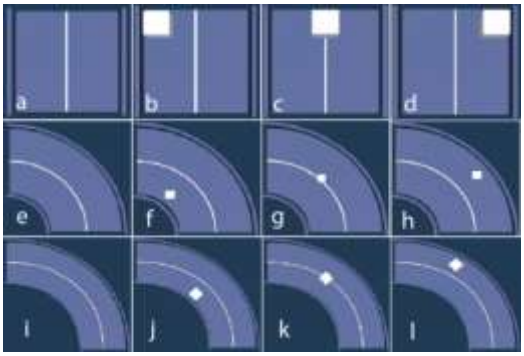
Pada Gambar 1 dijelaskan sembilan posisi sensor *raycast* yang diletakkan pada sudut 0°, 30°, 45°, 60°, 90°, -30°, -45°, -60°, dan -90°. Sensor *raycast*

berguna sebagai masukan untuk mengobservasi lingkungan NPC apakah terdapat tembok atau tidak. Sensor *raycast* dapat mendeteksi tembok apabila sensor *raycast* membentur tembok. Untuk mengetahui seberapa jauh jarak tembok dengan NPC dapat menggunakan *hitInfo distance* yang mana *hitInfo distance* adalah jarak NPC dengan tembok seperti yang dapat dilihat pada Gambar 2.




**Gambar 2**. (a) *raycast distance*, (b) *hitInfo*

Setelah melakukan perancangan sensor *raycast*, maka dilakukan perancangan skenario untuk menggambarkan skenario apa saja yang digunakan dalam proses pelatihan NPC. Terdapat 12 skenario yang digunakan dalam proses pelatihan. Semua skenario pelatihan dapat dilihat pada Gambar 3.




**Gambar 3**. Skenario Pelatihan NPC

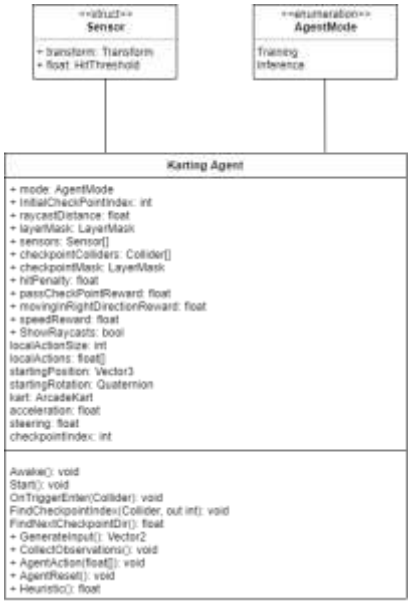
Pada Gambar 3 digambarkan semua skenario pelatihan NPC. Terdapat empat skenario pada trek lurus yaitu skenario a,b,c dan d. Pada skenario ini bertujuan agar NPC dapat melaju pada trek lurus tanpa *obstacle*, pada trek lurus dengan *obstacle* di sisi kiri, pada trek lurus dengan *obstacle* di sisi tengah, dan pada trek lurus dengan *obstacle* di sisi kanan. Kemudian, terdapat empat skenario pada trek tikungan pendek yaitu skenario e, f, g, dan h. Pada skenario ini bertujuan agar NPC dapat melaju pada trek tikungan pendek tanpa *obstacle,* pada trek tikungan pendek dengan *obstacle* di sisi kiri, pada trek tikungan pendek dengan *obstacle* di sisi tengah, dan pada trek tikungan pendek dengan *obstacle* di sisi kanan. Terakhir, terdapat empat skenario pada trek tikungan panjang yaitu skenario i, j, k, dan l. Pada skenario ini bertujuan agar NPC dapat melaju pada trek tikungan panjang tanpa *obstacle,* pada trek tikungan panjang dengan *obstacle* di sisi kiri, pada trek tikungan panjang dengan *obstacle* di sisi tengah, dan pada trek tikungan panjang dengan *obstacle* di sisi kanan.

Setelah melakukan perancangan skenario, maka dilakukan perancangan diagram alir kerja NPC yang dapat dilihat pada Gambar 4.


**Gambar 4**. Diagram Alir Kerja NPC

Pada Gambar 4 dijelaskan diagram alir kerja dari NPC yaitu pertama NPC akan melakukan observasi lingkungan menggunakan sensor raycast. Hasil observasi dari sensor raycast akan diproses menjadi informasi. Setelah itu NPC akan melakukan aksi berdasarkan informasi yang telah didapatkan.

Terakhir, dilakukan perancangan *class diagram*. Hasil perancangan *class diagram* dapat dilihat pada Gambar 5.




**Gambar 5**. Diagram Kelas *Karting Agent*

Pada Gambar 5 dijelaskan tentang *class* yang digunakan untuk menerapkan NPC pada Karting Microgame menggunakan UnityML-Agents. Pada *class* Karting Agent terdapat *class struct* sensor yang

memiliki 2 atribut *public* yaitu *transform* dan *HitThreshold*. Selain itu, terdapat *enumeration* AgentMode yang berisikan 2 enum yaitu *Training* dan *Inference.* Pada *class* Karting Agent memiliki 12 atribut *public* dan 8 atribut tanpa *identifier*. Semua atribut yang telah digambarkan pada *class diagram* digunakan untuk mendukung jalannya aksi-aksi yang terdapat pada *class* Karting Agent. Aksi-aksi tersebut adalah *awake, start, OnTriggerEnter, FindCheckPointIndex* yang bersifat *void* serta terdapat 3 aksi yang memiliki *identifier public* dan bersifat *void* yaitu *CollectObservation, AgentAction,* dan *AgentReset*. Selain itu terdapat 3 aksi yang memiliki *return value* yaitu *FindNextCheckpointDir* dan *Heuristic* dengan *return value float* dan *GenerateInput* dengan *return value Vector2.*

### 2.3. Implementasi

Implementasi dilakukan pada *game object karting agent* dengan menambahkan *script KartingAgent* dan *Behavior Parameter*. *Script Karting Agent* berfungsi untuk melakukan observasi lingkungan dan menjalankan aksi sesuai dengan hasil observasi. *Behavior Parameters* berperan sebagai *script* yang mengatur *brain properties* dan pengaturan *behavior* NPC. Implementasi yang telah dilakukan dapat dilihat pada Gambar 6.


**Gambar 6.** Implementasi Game Object Karting Agent pada inspektor unity

### 2.4. Pengujian

Pengujian dibagi menjadi dua yaitu pengujian performa FPS dan pengujian performa NPC. Pengujian performa FPS dilakukan dengan cara membandingkan FPS sebelum dan sesudah menerapkan NPC yang telah dikembangkan. Pengujian performa FPS penting untuk dilakukan karena FPS merupakan faktor yang menentukan sebuah performa gim [14]. Sebuah gim dapat dikatakan *playable* apabila dapat berjalan pada FPS minimal yaitu 30 FPS [15]. Setelah melakukan pengujian performa FPS, dilakukan pengujian performa NPC. Pengujian performa NPC dilakukan dengan cara mencatat berapa lama waktu yang dibutuhkan oleh NPC untuk melintasi trek dibandingkan dengan waktu pemain untuk melintasi trek pada berbagai macam trek.

## 3. HASIL DAN PEMBAHASAN

Hasil dari pengujian performa FPS dan pengujian performa NPC akan dijelaskan lebih lanjut pada subbab 3.1 dan subbab 3.2.

### 3.1. Pengujian Performa FPS

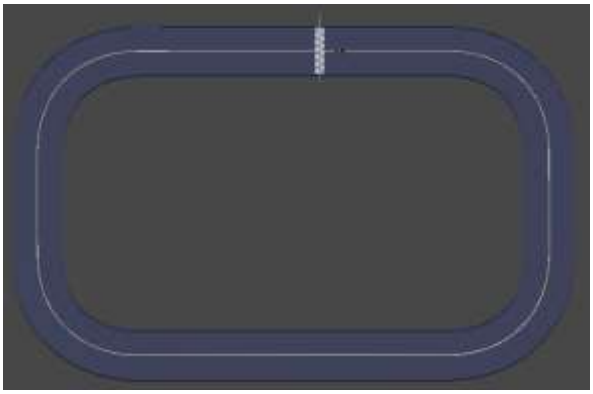
Pengujian performa FPS dilakukan untuk menguji performa dari sebuah sistem yang sudah dibangun. Pengujian performa dilakukan dengan cara membandingkan *frame per second* (FPS) sebelum dan sesudah menerapkan NPC yang telah dikembangkan menggunakan Unity ML-Agents. Hasil pengujian performa FPS dapat dilihat pada Tabel 1.

**Tabel 1.** Hasil Pengujian Performa FPS

| Sebelum (FPS) | Sesudah (FPS) |
|---|---|
| 290 | 286 |

### 3.2. Pengujian Performa NPC

Pengujian performa NPC dilakukan untuk mengetahui performa NPC dalam melintasi trek dengan cara mencatat berapa lama waktu yang dibutuhkan oleh NPC untuk melintasi trek dibandingkan dengan waktu pemain untuk melintasi trek pada berbagai macam trek. Hasil pengujian performa NPC yang telah dilakukan adalah hampir pada setiap trek NPC selalu mendapatkan waktu terbaik yang lebih cepat dan waktu total yang lebih sedikit untuk menempuh 3 lap putaran dibandingkan pemain. NPC berhasil mengungguli performa pemain pada trek 1 tanpa *obstacle* pada gambar 7, trek 1 dengan *obstacle* pada gambar 8, trek 2 tanpa *obstacle* pada gambar 9, trek 2 dengan *obstacle* pada gambar 10, trek 3 tanpa *obstacle* pada gambar 11, dan trek 3 dengan *obstacle* pada gambar 12. Performa NPC pada trek 2 tanpa *obstacle* hanya kalah 1 detik lebih lambat saja dibandingkan dengan pemain. Semua

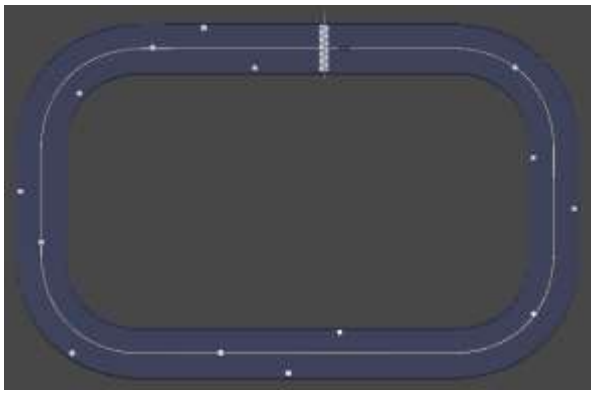


**Gambar 7**. Trek 1 tanpa *obstacle*

hasil pengujian performa NPC dapat dilihat pada Tabel 2 sampai dengan 8.

**Tabel 2.** Pengujian performa NPC pada trek 1 tanpa *obstacle*

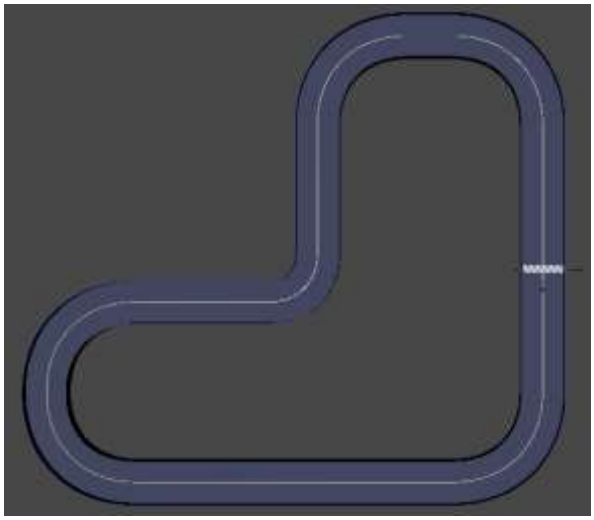
| Lap | Jumlah Waktu NPC (detik) | Jumlah Waktu Pemain (detik) |
|---|---|---|
| 1 | 19,1 | 19,6 |
| 2 | 18,4 | 18,6 |
| 3 | 18,4 | 18,9 |
| Waktu terbaik (detik) | 18,4 | 18,6 |
| Waktu total (detik) | 55,9 | 57,1 |



**Gambar 8**. Trek 1 dengan *obstacle*

**Tabel 3.** Pengujian performa NPC pada trek 1 dengan *obstacle*

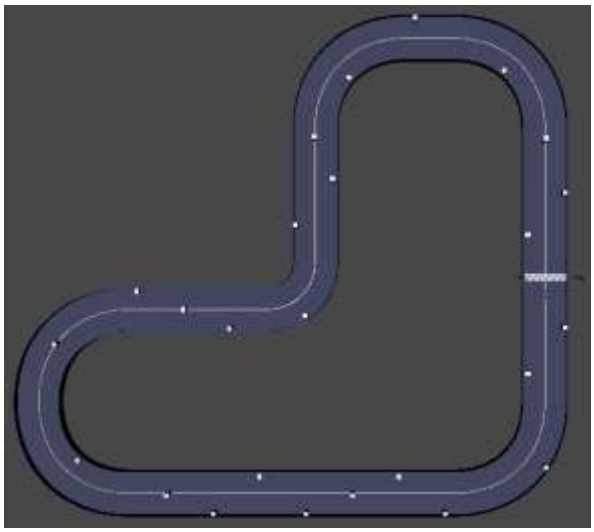
| Lap | Jumlah Waktu NPC (detik) | Jumlah Waktu Pemain (detik) |
|---|---|---|
| 1 | 19,1 | 20,5 |
| 2 | 18,4 | 19,5 |
| 3 | 18,4 | 19,4 |
| Waktu terbaik (detik) | 18,4 | 19,4 |
| Waktu total (detik) | 55,9 | 59,4 |



**Gambar 9**. Trek 2 tanpa *obstacle*

**Tabel 4.** Pengujian performa NPC pada trek 2 tanpa *obstacle*

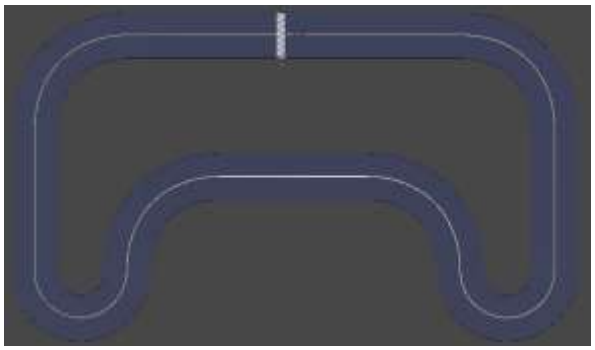
| Lap | Jumlah Waktu NPC (detik) | Jumlah Waktu Pemain (detik) |
|---|---|---|
| 1 | 24,9 | 25,1 |
| 2 | 24,1 | 24,0 |
| 3 | 24,1 | 23,8 |
| Waktu terbaik (detik) | 24,1 | 23,8 |
| Waktu total (detik) | 73,1 | 72,1 |



**Gambar 10**. Trek 2 dengan *obstacle*

**Tabel 5.** Pengujian performa NPC pada trek 2 dengan *obstacle*

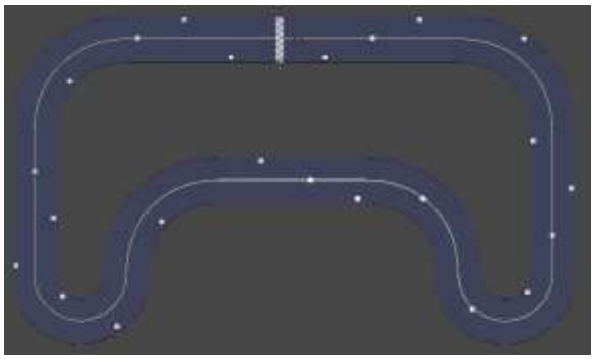
| Lap | Jumlah Waktu NPC (detik) | Jumlah Waktu Pemain (detik) |
|---|---|---|
| 1 | 24,8 | 26,1 |
| 2 | 24,1 | 24,6 |
| 3 | 24,1 | 24,8 |
| Waktu terbaik (detik) | 24,1 | 24,6 |
| Waktu total (detik) | 73 | 75,5 |



**Gambar 11**. Trek 3 tanpa *obstacle*

**Tabel 6**. Pengujian performa NPC pada trek 3 tanpa *obstacle*

| Lap | Jumlah Waktu NPC (detik) | Jumlah Waktu Pemain (detik) |
|---|---|---|
| 1 | 23,3 | 23,9 |
| 2 | 22,5 | 22,2 |
| 3 | 22,5 | 22,9 |
| Waktu terbaik (detik) | 22,5 | 22,2 |
| Waktu total (detik) | 68,3 | 69 |



**Gambar 12**. Trek 3 dengan *obstacle*

**Tabel 7**. Pengujian performa NPC pada trek 3 dengan *obstacle*

| Lap | Jumlah Waktu NPC (detik) | Jumlah Waktu Pemain (detik) |
|---|---|---|
| 1 | 23,5 | 26,2 |
| 2 | 22,8 | 23,3 |
| 3 | 22,8 | 23,2 |
| Waktu terbaik (detik) | 22,8 | 23,2 |
| Waktu total (detik) | 69,1 | 72,7 |

## 4. KESIMPULAN

Berdasarkan serangkaian tahap penelitian yang telah dilakukan. Dapat diambil kesimpulan berdasarkan hasil pengujian performa FPS hanya terdapat sedikit penurunan FPS yaitu dari 290 menjadi 286 FPS yang mana masih tergolong *playable* dikarenakan masih memiliki total FPS di atas 30 FPS. Berdasarkan hasil pengujian performa NPC dibandingkan dengan pemain manusia didapatkan hasil bahwa performa NPC hampir selalu mendapatkan hasil waktu terbaik yang lebih cepat dan waktu total yang lebih sedikit untuk menempuh 3 lap putaran dibandingkan dengan pemain.

## 5. DAFTAR PUSTAKA


[1] "A Visual History of Racing Games - IGN", *IGN*, 2015. [Online]. Available: https://www.ign.com/articles/2015/03/27/a-visual-history-of-racing-games. [Accessed: 22- Jan- 2020].

[2] M. Chan, C. Chan and C. Gelowitz, "Development of a Car Racing Simulator Game Using Artificial Intelligence Techniques", *International Journal of Computer Games Technology*, vol. 2015, pp. 1-6, 2015.

[3] A. Beech, "Asphalt 9: Legends release date and Livestream! | Gameloft Central", *Gameloft Central*, 2018. [Online]. Available: http://www.gameloft.com/central/asphalt/asphalt-9-legends/asphalt-9-legends-livestream-and-release-date. [Accessed: 22- Jan- 2020].

[4] "IR Information : Sales Data - Top Selling Title Sales Units", *Nintendo Co., Ltd.*, 2019. [Online]. Available: https://www.nintendo.co.jp/ir/en/finance/software/index.html. [Accessed: 22- Jan- 2020].

[5] A. Kim and J. Bae, "Development of Mobile Game Using Multiplatform (Unity3D) Game Engine", p. 2, 2014. [Accessed 30 November 2020].

[6] Unity Technologies "Karting Microgame | Templates | Unity Asset Store", *Assetstore.unity.com*, 2019. [Online]. Available: https://assetstore.unity.com/packages/templates/karting-microgame-150956. [Accessed: 22-

Jan- 2020].

[7] J. Wang and Y. Lin, "Game AI: Simulating Car Racing Game by Applying Pathfinding Algorithms", *International Journal of Machine Learning and Computing*, pp. 13-18, 2012. Available: 10.7763/ijmlc.2012.v2.82 [Accessed 22 January 2020].

[8] A. El Rhalibi, K. Wong and M. Price, "Artificial Intelligence for Computer Games", *International Journal of Computer Games Technology*, vol. 2009, pp. 1-3, 2009. Available: 10.1155/2009/251652.

[9] A. Juliani et al., "Unity: A General Platform for Intelligent Agents", *arXiv.org*, 2018. [Online]. Available: http://arxiv.org/abs/1809.02627. [Accessed: 22- Jan- 2020].

[10] S. Sehad and C. Touzet, "Reinforcement learning and neural reinforcement learning," *ESANN 94 Ed. M Verleysen DFacto Publ. Bruxelles*, pp. 1–6, 1994.

[11] H. Teigar, M. Storožev and J. Saks, "2D Racing game using reinforcement learning and supervised learning", p. 1, 2018.

[12] F. G. Glavin and M. G. Madden, "Learning to shoot in first person shooter games by stabilizing actions and clustering rewards for reinforcement learning," 2015 IEEE Conf. Comput. Intell. Games, CIG 2015 - Proc., pp. 344–351, 2015.

[13] S. Wender and I. Watson, "Applying reinforcement learning to small scale combat in the real-time strategy game StarCraft:Broodwar", *2012 IEEE Conference on Computational Intelligence and Games (CIG)*, 2012. Available: 10.1109/cig.2012.6374183 [Accessed 22 January 2020].

[14] M. Claypool and K. Claypool, "Perspectives, frame rates and resolutions: It's all in the game," FDG 2009 - 4th Int. Conf. Found. Digit. Games, Proc., pp. 42–49, 2009.

[15] S. Stewart, "What Is The Best FPS For Gaming? [2020 Guide] - GamingScan", *GamingScan*, 2020. [Online]. Available: https://www.gamingscan.com/best-fps-gaming. [Accessed: 22- Sep- 2020].