Journal of Soft Computing Exploration

Homepage: htttps://shmpublisher.com/index.php/joscex

p-ISSN: 2746-7686
e-ISSN: 2746-0991

.

Implementation of a reinforcement learning system with deep q network algorithm in the amc dash mark i game

Wargijono Utomo[1]
[1]Department of Information System, Universitas Krisnadwipayana, Indonesia

Article Info

Article history:

Received December 8, 2023
Revised December 15, 2023
Accepted March 4, 2024

Keywords:

Reinforcement learning
Deep q network
Amc dash mark I
Game

ABSTRACT

Reinforcement learning is a branch of artificial intelligence that trains algorithms using a trial-and-error system. Reinforcement learning interacts with its environment and observes the consequences of its actions in response to rewards or punishments received. Reinforcement Learning uses information from every interaction with its environment to update its knowledge. The problem identified from this research is the lack of consistency, which is not always the same for Non-Player Characters (Agents) in the process of exploring an environment (Game environment). This research uses the Software Development Life Cycle (SDLC) Waterfall model method to train Non Player Characters (Agents) in the Amc Dash Mark I Game which uses the Deep Q Network (DQN) algorithm in several stages. Training results show improvements in model performance over time. The average duration of the episode and average reward episode showed an increase of 7.75 to 24.7, while the exploration rate decreased to 0.05. This indicates that the model has experienced learning and is improving to achieve better rewards by performing fewer actions. The lower loss also shows that the model has succeeded in reducing prediction errors and improving prediction capabilities.

Corresponding Author:

Wargijono Utomo,
Department of Information System,
Universitas Krisnadwipayana, Indonesia
Email: uwargiono@gmail.com
https://doi.org/10.52465/joscex.v5i1.271

1. INTRODUCTION

In the beginning, computers were only used as calculating machines. However, as time passes, the role of computers is increasingly dominant in human life. Now, computers are not only used as calculating machines, but are also expected to be able to do all the tasks that humans can do. Technological developments are increasingly rapid [1], with new innovations continuing to emerge in various aspects of life, including different fields. Technology is something that needed by human [2]. Not only is it used to make human work easier, technology also continues to develop in the entertainment sector, such as games that are currently increasingly popular and continue to develop rapidly [3]. The emergence of online gaming through the internet has gained widespread popularity and addiction among individuals, particularly teenagers [4].

A game or game is a fun medium that does not make people who play it feel bored [5]. Games have become an important part of the lives of young teens and adults, both boys and girls, from various social and economic backgrounds. Games are available at various places and are often played by teenagers; up to half of all teenagers report having played a game in the previous day. Those who play it regularly usually play it for

at least one hour per day [6]. In games, artificial intelligence is often used to challenge players. This challenge can be an opportunity to fight against a machine that has certain intellectual abilities in thinking, so players don't need to look for other opponents if they want to play. An efficient and effective algorithm must support the thinking skills applied in the game, taking into account the applicable rules [7]. One of the algorithms commonly used in games is the Deep Q Network (DQN) algorithm. The correlation between DQN and artificial intelligence lies in the way DQN learns and makes decisions in dynamic environments, such as games. DQN adapts the concept of reinforcement learning to allow agents (in this case, usually NPC characters in games) to learn from interactive experiences with the game environment.

The field of artificial intelligence (AI) is a part of computer science that aims to make machines, such as computers, have the ability to complete tasks with intelligence or even exceed human capabilities in some aspects. In game development, one of the challenges that arises is creating a control system for non-player character (NPC) or Agent characters that is capable of realistic and responsive behavior. This agent is an autonomous entity in a virtual world, such as in a game or virtual reality, that plays the role of a character in a story or game scenario. The main challenge in this case is giving Agents the ability to perform improvised actions, which differentiates them from characters in animated films that have rigidly programmed actions, or "avatars" in games that are controlled directly by the player in real time [8].

The development movement of agents manually requires quite a lot of time and effort, especially if there are a lot of agents in the game. In recent years, reinforcement learning has become one of the techniques used to overcome this problem [9]. Reinforcement learning is a branch of artificial intelligence that trains algorithms using a trial and error system [10]. Reinforcement learning interacts with its environment and observes the consequences of its actions in response to rewards or punishments received [11]. Reinforcement learning uses information from every interaction with its environment to update its knowledge [12]. Meanwhile, according to [13] Reinforcement learning is a type of machine learning where agents learn something by performing certain actions and seeing the results of these actions and trying to maximize the rewards received through interaction with the environment in the form of rewards with negative or positive values [14]. According to previous research, several algorithms, such as Deep Deterministic Policy Gradient (DDPG) and Duel DQN, can be implemented for this game and extensively compared with the aforementioned results [12], [15]. Another important observation is that the speed of obstacles increases as the score increases, so the game can be divided into stages. Reinforcement learning models are generally called agents because they learn to behave in a given environment by executing random actions and getting rewarded after each. Each stage receives input from the previous stage but is trained individually [10], [11]. Gap analysis highlights the lack of realistic movement control in NPC characters and a less adaptive response to the game environment. This research sharpens the focus on applying the reinforcement learning method with the Deep Q Network algorithm to overcome these weaknesses, making NPC movement control more intelligent, responsive, and adaptive in the AMC Dash Mark I game. This provides an important innovation in providing a more immersive and immersive gaming experience. challenging for the players. The uniqueness of this research is the approach used to overcome the NPC movement control problem in the AMC Dash Mark I game. The use of the Deep Q Network algorithm and the application of reinforcement learning provide a strong foundation for producing NPCs that are more adaptive, responsive, and able to learn the game environment better. well, which perhaps hasn't been done so clearly before in the game. The goal in this research is to further improve the performance of the Reinforcement Learning algorithm and reduce the variance in scores.

2. METHOD

In this research, it can be illustrated in Figure 1 as the stages that must be passed to complete the research.

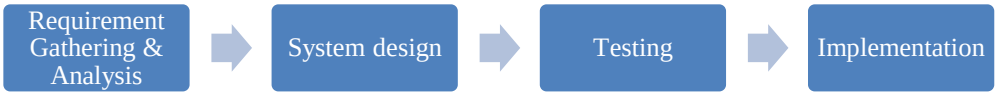

Figure 1. Research methodology

This research uses secondary methods that utilize existing data, namely those sourced from https://github.com/nicknochnack/DinoAI/tree/main. The approaches used include requirements gathering, system design, testing, and implementation which can be explained as follows [3], [16]:

1) Requirements gathering is the process of identifying, understanding, and documenting the needs that must be met by a software system or project.
2) System design is the process of designing the structure, components, and features of a software system or information system as a whole. The goal of system design is to produce a clear and structured plan for how the system will function, interact, and meet the needs of users and stakeholders.

3) Testing is an important process in developing software or information systems, which aims to evaluate the quality, performance, and reliability of the system. Testing is carried out to ensure that the system functions as expected, meets user requirements, and works well in various situations and conditions.
4) Implementation is the stage in the software or information system development cycle where the previously designed system design is converted into executable program code.

Game

Games are a form of recreational or competitive activity that usually involves players in an interactive environment, either virtually (video games) or in real life. Games have certain rules, goals that must be achieved, and often offer challenges or entertainment. The goals expressed in games can vary, including education, entertainment, and simulation [17]. In the context of video games, there are various genres such as action games, adventure, strategy games, RPG (role), puzzle games, and many more. Each genre usually has different rules, game mechanics, and goals. The main goal of the game is to please the player, although some games also provide challenging challenges or specific goals that must be achieved. Most games rely on rules and interactions between the player or players and their virtual environment to create an entertaining or educational experience [18].

Reinforcement Learning

Reinforcement Learning (RL) is a machine learning paradigm that studies how agents can perform actions in an environment to achieve certain goals that maximize rewards [19]. The reinforcement learning process occurs through iteration in which the agent performs actions in the environment, receives feedback (rewards) for these actions, and gradually updates the action decisions taken based on the experience gained from interactions with the environment [14]. The aim of reinforcement learning is to develop optimal policies, namely rules or strategies that allow agents to take action to maximize the total reward received in the long term [15].

Deep Q Network Algorithm

The Deep Q Network (DQN) is a reinforcement learning algorithm developed to overcome complex problems in machine learning [20]. This algorithm is a combination of reinforcement learning with a deep artificial neural network [21]. The essence of the DQN algorithm is to utilize an artificial neural network to model the Q function, which estimates action values in an environment or game based on states [22]. This Q value measures how profitable an action is in a particular situation. This is represented in the following equation [23]:

$$Y_k = r + \gamma max_{a'} Q(s', a'; \theta_k^-) \quad (1)$$

AMC Dash Mark I

Amc Dash Mark I is a platformer game that takes reference from the geometry dash game as seen in Figure 2 Geometry dash. This game has a faster rhythm, and the agent being played is a square with the main goal of reaching the end of the level without colliding with any obstacles [12]. This game requires precision and timing to jump, bounce, and avoid various obstacles such as sharp obstacles, snares, and dangerous holes [24]. The Geometry Dash game offers a variety of game modes, including normal mode, where the player must complete the level in the correct way, and practice mode, which allows the player to practice and learn obstacle patterns before attempting the real level [25].

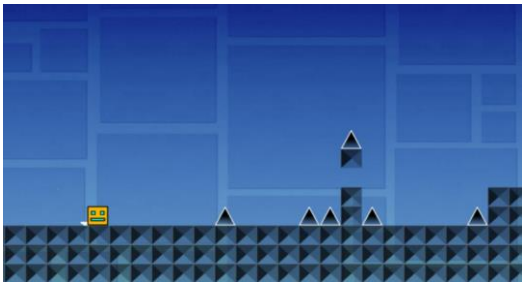

Figure 2. Geometry dash

3. RESULTS AND DISCUSSIONS

Requirements Gathering & Analysis

The problem with this research is the lack of consistency, which is not always the same for non-player Characters (Agents) in the process of exploring an Environment (Game environment).

System Design

In the system design section, a custom environment will be created that allows Non-Player Characters (Agents) to interact with their environment.

Creating the AMC dash mark i game

The following is the process of making the AMC Dash Mark I Game, which goes through several stages including: First Import Game Library: In the first process of creating the Amc Dash Game, several libraries are imported, as seen in figure 3.a Import game library. The code in Figure 3.a imports all the libraries needed to develop the Game using pyGame and performs CSV data manipulation if needed [26]. Second Main Player Class: The code in figure 3.b PyGame Configuration is initialization and some initial configuration for developing a game using pyGame [27].

```
import csv
import os
import random
import numpy as np
import pygame
from pygame.math import Vector2
from pygame.draw import rect
```

(a)

```
pygame.init()
screen = pygame.display.set_mode([950, 600])
done = False
start = False
clock = pygame.time.Clock()
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255)
color = lambda: tuple([random.randint(0, 255) for i in range(3)])
GRAVITY = Vector2(0, 0.86)
```

(b)

Figure 3. a. Import game library and b. Configuration

Third Obstacle Class: The code in Figure 4.a defines the Draw class, which is a subclass of pyGame.sprite.Sprite. This class is responsible for drawing objects in the game. Fourth Global Variables: The code given in Figure 4.b is part of initializing several variables that will be used in the game using the pyGame module.

```
class Draw(pygame.sprite.Sprite):
    def __init__(self, image, pos, *groups):
        super().__init__(*groups)
        self.image = image
        self.rect = self.image.get_rect(topleft=pos)
```

(a)

```
font = pygame.font.SysFont("lucidaconsole", 20)
avatar = pygame.image.load(os.path.join("images", "avatar.png"))
pygame.display.set_icon(avatar)
alpha_surf = pygame.Surface(screen.get_size(), pygame.SRCALPHA)
player_sprite = pygame.sprite.Group()
elements = pygame.sprite.Group()
spike = pygame.image.load(os.path.join("images", "obj-spike.png"))
spike = resize(spike)
coin = pygame.image.load(os.path.join("images", "coin.png"))
coin = pygame.transform.smoothscale(coin, (32, 32))
block = pygame.image.load(os.path.join("images", "block_1.png"))
block = pygame.transform.smoothscale(block, (32, 32))
orb = pygame.image.load((os.path.join("images", "orb-yellow.png")))
orb = pygame.transform.smoothscale(orb, (32, 32))
trick = pygame.image.load((os.path.join("images", "obj-breakable.png")))
trick = pygame.transform.smoothscale(trick, (32, 32))
```

(b)

Figure 4. a. Class draw and b. Global variables

Creating a reinforcement learning model

After creating the game that you want to test, the process is then carried out to create a reinforcement learning model that will be used to train the Game that was created previously. The steps for creating a Reinforcement Model are as follows. Import Library, in the first process of system design, here we will import several libraries, as seen in Figure 5.a. The code only contains library imports and does not contain implementation or special functions of the library. In the second stage, Create an Environment In this stage, a custom environment is created that has the task of capturing the screen, as seen in Figure 5.b. The codes in Figure 5.b are used to initialize the game environment by setting the observation space, the action space, and other necessary parameters, such as the location of the screenshot and the game in the screenshot.

```python
from mss import mss
import pydirectinput
import cv2
import numpy as np
import pytesseract
from matplotlib import pyplot as plt
import time
from gym import Env
from gym.spaces import Box, Discrete
```

(a)

```python
class WebGame(Env):
    def __init__(self):
        super().__init__()
        self.observation_space = Box(low=0, high=255, shape=(1,83,100), dtype=np.uint8)
        self.action_space = Discrete(2)
        self.cap = mss()
        self.game_location = {'top': 250, 'left': 50, 'width': 350, 'height': 500}
        self.done_location = {'top': 280, 'left': 90, 'width': 500, 'height': 50}
```

(b)

Figure 5. a. Import library and b. Class web game

In the third stage of creating a Deep Q Network Model in creating a DQN model, several libraries are imported from Stable Baseline 3 as seen in Figure 6. The code in Figure 6, is the import and use of several modules in the "stable_baselines3" library to manage file paths and create custom callbacks to save the model during training as well as for the implementation of the Deep Q-Network (DQN) algorithm in reinforcement learning.

```python
import os
from stable_baselines3.common.callbacks import BaseCallback
from stable_baselines3 import DQN
from stable_baselines3.common.monitor import Monitor
from stable_baselines3.common.vec_env import DummyVecEnv, VecFrameStack
```

Figure 6. Import library stable baselines3

Testing

In the testing process, the non-player character (Agent) will be trained to pass through obstacles that have been created in the AMC Dash Mark I Game. At this stage, the game is placed in a screenshot area that has been previously designed, as seen in Figure 7. marked with a dotted line. broken off in green.

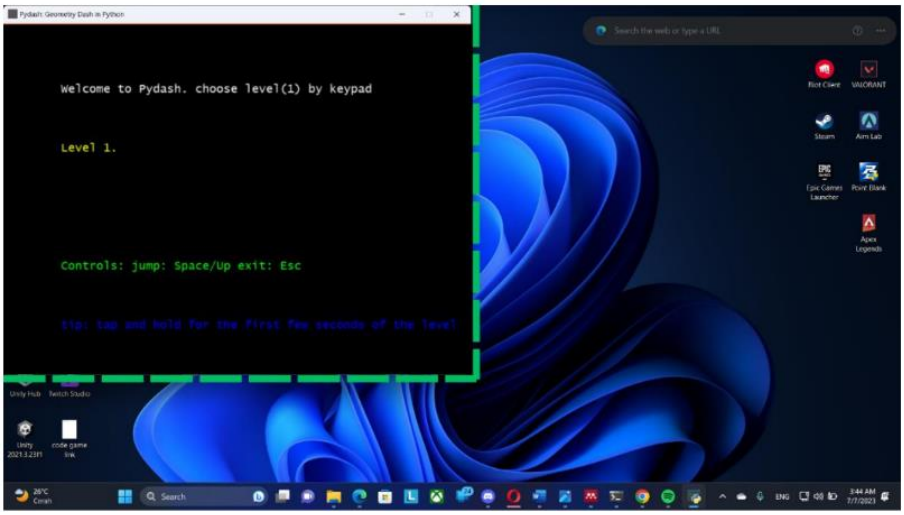

Figure 7. Game positioning

Implementation

After carrying out the system design, the implementation stage is then carried out, where the system that has been created will be implemented in the Amc Dash Mark I Game to get the results of tests that will be carried out on Non-Player Characters (Agents) in the Game. After the testing process, the system is then implemented into the AMC Dash Mark I game by calling the code in Figure 8.

```
model = DQN.load(os.path.join('train/best_model_8000.zip'))
```

Figure 8. Best model

The code in Figure 8. will reflect the final state of the DQN model when it stops training after being trained for 8000 iterations (epochs) or episodes. Therefore, this model can be used to continue further training or used for other purposes such as evaluating performance, policies, or strategies that have been learned by the model. Next, the non-player character (Agent) will walk past the spikes and blocks as seen in Figure 9.

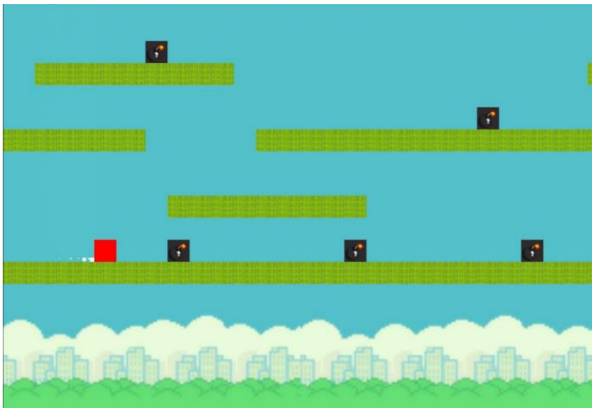

Figure 9. Amc dash mark I

Then the results obtained by the non-player character (Agent) while passing through the spikes and blocks are shown in Figure 9.

Figure 10. Total reward

The following is an explanation for each episode which can be seen in Figure 10:

1) Episode 0: The total reward obtained in this episode is 12. In this episode, the model has taken 1 random action. A high total reward indicates that the model succeeded in achieving good rewards in this episode.
2) Episode 1: The total reward obtained in this episode is 8. In this episode, the model also takes one random action. Although the total rewards were slightly lower than in the previous episode, this could still be considered a good performance.
3) Episode 2: The total reward obtained in this episode is 12. In this episode, the model no longer performs random actions (Z=0). The model may have used the policies it learned during training and succeeded in achieving good rewards.
4) Episode 3: The total reward obtained in this episode is 12. In this episode, the model has taken 1 random action. Despite this, the model still achieves a high total reward as in the previous episode.

5) Episode 4: The total reward obtained in this episode is 5. In this episode, the model no longer performs random actions (Z=0). A lower total reward may indicate that the model is having difficulty in the given task or that there is still room for improvement in the policies it has learned.

In experiments carried out for the implementation of the reinforcement learning system using the deep q network algorithm in the amc dash mark i game, a significant increase in the performance of NPC movement control was seen. The implementation of Reinforcement Learning using the Deep Q Network (DQN) algorithm in the AMC Dash Mark I game succeeded in increasing the NPC's ability to adapt to the game environment, provide smarter responses, and show clear progress over time.

4. CONCLUSION

This research focuses on the application of the reinforcement learning system with the Deep Q Network algorithm in the AMC Dash Mark I game. The results show an increase in the consistency and performance of Non-Player Characters (NPC) in the game. The application of the SDLC (Software Development Life Cycle) waterfall model to train NPCs shows a gradual increase in the quality of NPC behavior, as evidenced by changes in average episode length, average reward episodes, and a decrease in exploration rate. This confirms that this learning model is effective in improving NPC performance.

Training results show improvements in model performance over time. The average duration of the episode and average reward episode showed an increase of 7.75 to 24.7, while the exploration rate decreased to 0.05. This indicates that the model has experienced learning and is improving to achieve better rewards by performing fewer actions. The lower loss also shows that the model has succeeded in reducing prediction errors and improving prediction capabilities. Then the results are implemented, and the results are obtained. The results show that the lowest total reward is 5 and the highest reward is 24. These results show that during the 100 experimental episodes, the non-player character (Agent) has not been consistent in taking more optimal actions based on the Q value given. has been studied.

The findings of this research provide a strong foundation for further development in optimizing NPC performance in games. The next steps could include using alternative algorithms such as the deep deterministic policy gradient (DDPG) or other methods to compare and improve NPC performance. Additionally, the further application to other games or adaptation of this model for more complex scenarios could be prospective next steps. Further possible applications involve the development of more adaptive, intelligent and responsive NPC systems in demanding gaming contexts.

REFERENCES

[1] A. A. Nurdin, G. N. Salmi, K. Sentosa, A. R. Wijayanti, and A. Prasetya, "Utilization of Business Intelligence in Sales Information Systems," *J. Inf. Syst. Explor. Res.*, vol. 1, no. 1, pp. 39–48, Dec. 2022, doi: 10.52465/joiser.v1i1.101.

[2] A. Dwiantoro, I. Maulana, N. P. Damayanti, and R. N. Al Zahra, "Artificial intelligence (AI) imaging for enhancement of parking security," *J. Student Res. Explor.*, vol. 1, no. 1, pp. 15–20, Dec. 2022, doi: 10.52465/josre.v1i1.110.

[3] M. Mustofa, J. L. Putra, and C. Kesuma, "Penerapan Game Development Life Cycle Untuk Video Game Dengan Model Role Playing Game," *Comput. Sci.*, vol. 1, no. 1, pp. 27–34, 2021, doi: 10.31294/coscience.v1i1.158.

[4] R. C. Gabrito, R. Y. I. Jr., and J. F. P. Velza, "Impact of Online Gaming on the Academic Performance of DEBESMSCAT-Cawayan Campus Students," *Sci. J. Informatics*, vol. 10, no. 4, pp. 423–434, 2023.

[5] R. J. Jordy, H. Marcos, J. Wijaya Kusuma, D. Intan Surya Saputra, and P. Purwadi, "Game design documents for mobile elementary school mathematic educative games," *J. Soft Comput. Explor.*, vol. 4, no. 2, May 2023, doi: 10.52465/joscex.v4i2.129.

[6] J. Arjoranta, "How to Define Games and Why We Need to," *Comput. Games J.*, vol. 8, no. 3–4, pp. 109–120, 2019, doi: 10.1007/s40869-019-00080-6.

[7] Y. Zhao, Y. Wang, Y. Tan, J. Zhang, and H. Yu, "Dynamic Jobshop Scheduling Algorithm Based on Deep Q Network," *IEEE Access*, vol. 9, pp. 122995–123011, 2021, doi: 10.1109/ACCESS.2021.3110242.

[8] T. Hazra and K. Anjaria, *Applications of game theory in deep learning: a survey*, vol. 81, no. 6. Multimedia Tools and Applications, 2022. doi: 10.1007/s11042-022-12153-2.

[9] R. Köster and M. J. Chadwick, "What can classic Atari video games tell us about the human brain?," *Neuron*, vol. 109, no. 4, pp. 568–570, 2021, doi: 10.1016/j.neuron.2021.01.021.

[10] A. Fawzi *et al.*, "Discovering faster matrix multiplication algorithms with reinforcement learning," *Nature*, vol. 610, no. 7930, pp. 47–53, 2022, doi: 10.1038/s41586-022-05172-4.

[11] J. Montalvo, Á. García-Martín, and J. Bescós, "Exploiting semantic segmentation to boost reinforcement learning in video game environments," *Multimed. Tools Appl.*, vol. 82, no. 7, pp. 10961–10979, 2023, doi: 10.1007/s11042-022-13695-1.

[12] K. Shao, Z. Tang, Y. Zhu, N. Li, and D. Zhao, "A Survey of Deep Reinforcement Learning in Video Games," no. 61573353, pp. 1–13, 2019.

.

[13] A. S. Dharma and V. Tambunan, "Penerapan Model Pembelajaran dengan Metode Reinforcement Learning Menggunakan Simulator Carla," *J. Media Inform. Budidarma*, vol. 5, no. 4, p. 1405, 2021, doi: 10.30865/mib.v5i4.3169.

[14] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: lessons we have learned," *Int. J. Rob. Res.*, vol. 40, no. 4–5, pp. 698–721, 2021, doi: 10.1177/0278364920987859.

[15] D. Marwah, S. Srivastava, A. Gupta, and S. Verma, "Chrome Dino Run using Reinforcement Learning CS7IS2 Project (2019-2020)," 2020.

[16] M. J. Gomez, J. A. Ruipérez-Valiente, P. A. Martínez, and Y. J. Kim, "Applying learning analytics to detect sequences of actions and common errors in a geometry game," *Sensors (Switzerland)*, vol. 21, no. 4, pp. 1–16, 2021, doi: 10.3390/s21041025.

[17] C. Marín-Lora, M. Chover, J. M. Sotoca, and L. A. García, "A game engine to make games as multi-agent systems," *Adv. Eng. Softw.*, vol. 140, no. September 2019, p. 102732, 2020, doi: 10.1016/j.advengsoft.2019.102732.

[18] M. R. Islam, R. Rahman, A. Ahmed, and R. Jany, "NFS: A Hand Gesture Recognition Based Game Using MediaPipe and PyGame," 2022.

[19] S. Gronauer and K. Diepold, *Multi-agent deep reinforcement learning: a survey*, vol. 55, no. 2. Springer Netherlands, 2022. doi: 10.1007/s10462-021-09996-w.

[20] A. Ardiansyah and E. Rainarli, "Implementasi Q-Learning dan Backpropagation pada Agen yang Memainkan Permainan Flappy Bird," *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 6, no. 1, pp. 1–7, 2017, doi: 10.22146/jnteti.v6i1.287.

[21] I. Agustian *et al*., "Robot Obstacle Avoidance Dengan Algoritma Q-Learning," *J. TEKTRIKA*, vol. 05, no. 02, pp. 61–70, 2020.

[22] Á. L. Valdivieso Caraguay, J. P. Vásconez, L. I. Barona López, and M. E. Benalcázar, "Recognition of Hand Gestures Based on EMG Signals with Deep and Double-Deep Q-Networks," *Sensors*, vol. 23, no. 8, pp. 1–18, 2023, doi: 10.3390/s23083905.

[23] M. H. Alabdullah and M. A. Abido, "Microgrid energy management using deep Q-network reinforcement learning," *Alexandria Eng. J.*, vol. 61, no. 11, pp. 9069–9078, 2022, doi: 10.1016/j.aej.2022.02.042.

[24] S. Cano, N. Araujo, C. Guzman, C. Rusu, and S. Albiol-Pérez, "Low-cost assessment of user experience through EEG signals," *IEEE Access*, vol. 8, pp. 158475–158487, 2020, doi: 10.1109/ACCESS.2020.3017685.

[25] A. B. Harisa and W. K. Tai, "Pacing-based Procedural Dungeon Level Generation: Alternating Level Creation to Meet Designer's Expectations," *Int. J. Comput. Digit. Syst.*, vol. 12, no. 1, pp. 401–416, 2022, doi: 10.12785/ijcds/120132.

[26] C. R. Harris *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020, doi: 10.1038/s41586-020-2649-2.

[27] Muhammad Romzi and B. Kurniawan, "Pembelajaran Pemrograman Python Dengan Pendekatan Logika Algoritma," *JTIM J. Tek. Inform. Mahakarya*, vol. 03, no. 2, pp. 37–44, 2020.