Developing a Dynamic Decision Making of an Enemy Character in a Tower Defense Game using Reinforcement Learning Technique in Unity ML-Agents

Ekapong Nopawong* and Rawinan Praditsangthong

College of Digital Innovation and Information Technology, Rangsit University, Pathum Thani, Thailand
*Corresponding author, E-mail: Ekapong.n@rsu.ac.th

Abstract

A tower defense game refers to a game that player have to protect a tower and attack an enemy character. The enemy characters will response to fight the players. However, the characters cannot decide to choose a new direction movement when occurring in a new situation. Thus, the objective of this research is to develop the dynamic decision making for an enemy character in a tower defense game using reinforcement learning technique in the Unity ML-Agents. The learning is designed as the lesson learning from the lesson zero to the lesson three called the curriculum learning. Therefore, agents need to learn from the curriculum learning by receiving a cumulative reward score and deleting scores from punishment when learning errors occur. The results show that the learning of lesson zero to lesson three has the growth rate from the cumulative reward score. Furthermore, the wrong decision making of the agents has the regressive rate between lesson zero to lesson three from the value loss score. These two scores mean that the enemy characters can dynamically decide to choose direction movement in an environment.

Keywords: *Dynamic decision making, Agents, Enemy character, Curriculum learning, Lesson, Reinforcement learning*

1. Introduction

In a decade, there are several types of games that are developed, such as strategy, turn-based strategy, simulation, social competition, and tower defense. Especially, tower defense games are a favorite game from most players, both the mobile platform and Personal Computer (PC) platform, including Virtual Reality (VR) platform. Moreover, the tower defense games led to apply in several areas such as education or heath care (Michailidis, Barcias, Charles, He, & Balaguer-Ballester, 2019; Gulec, Yilmaz, Isler, O'Connor, & Clarke, 2018). The mechanics of tower defense games refer to attacking objects or items of the enemy and protecting the player's tower from an aggressor for one or more towers. The tower defense games consist of characters that will have a battle according to the specification qualification of each character. Thus, players have to choose the character; they want to play, such as a fighter, an archer, or a mage. All roles, both players and enemies, are created to have a difference in the specification feature of a battle (Philipa, Julian, & Elvis, 2014).

Therefore, a character movement needs to define the right direction and navigation into the environment. The character movement or character navigation is defined direction as a linear movement from point to point in a static environment. Commonly, character navigation uses a collider component to check the collision between two objects, but this component cannot detect the crash of the fast object. Then, there is the development of the navigation component called NavMesh. Nav stands for Navigation, and Mesh refers to parts of vertices, edges, and faces (He, Shi, & Li, 2016). Thus, NavMesh is a surface declaration to identify a restricted area; artificial intelligence (AI) systems are aware of direction and movement in the restricted area. The character slowly moves to a player for fighting, but it cannot avoid an object that comes in collide with itself. NavMesh can improve the mobility of the character in a static environment, but it cannot be the ability of the character movement in a dynamic environment. A player can fight the enemy quickly. These caused the players to lack excitement and balance in playing a game. This problem affects enjoyment and excitement for players. Consequently, most game engines, software development platform, use emerging technology to improve a dynamic environment, such as machine learning (ML) and artificial intelligence (AI).

The machine learning consists of three types: supervised learning, unsupervised learning, and reinforcement learning. In the game engine, reinforcement learning led to apply for the teaching of a

character. The concept of reinforcement learning refers to give cumulative reward and punishment in a situation; it is very interesting in the game development industry. The characters or the agents will have learning from trial and error of direction and movement in the environment. When the characters have the wrong direction and movement, they will be punished by decreasing cumulative reward score. On the other hand, if the characters move in the right direction and action; they will receive a cumulative reward score. Therefore, the characters will learn the right decision making in discover of direction, action, and movement from the cumulative reward. Reinforcement learning differs from supervised learning and unsupervised learning. Supervised learning has the answer attribute for training data, then the model is trained using the correct answer attribute. Unsupervised learning doesn't have the answer attribute, but learning appears from grouping data. On the other hand, reinforcement learning has no right answer; the characters will decide to choose the best action for each behavior. Reinforcement learning concept is presented in figure 1.

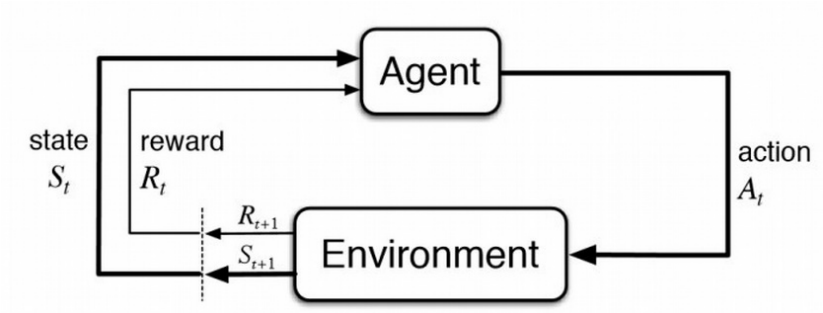

Figure 1 Reinforcement Learning Concept (Sutton & Barto, 2018)

From mentioned above, one interesting game engine is Unity, a cross-platform game engine. The Unity engine has developed the Software Development Kit (SDK) for machine learning using reinforcement learning through Python Application Programming Interface (API) called ML-Agents. The component of ML-Agents composes of learning environment, academy, brains, and agents (Arther, Vencent-Pierre, Esh, Yuan, Hunter, & Marwan, 2018). The learning environment is the game situation setting of the enemy character. The academy works with the agents to introduce decision-making processes; one academy is in one game. The agents are responsible for exploring and observing the environment and taking action. The final component is the brains; it receives from a result of the learning of the agents and defines as a policy for decision making. All components of ML-Agents are presented in figure 2.

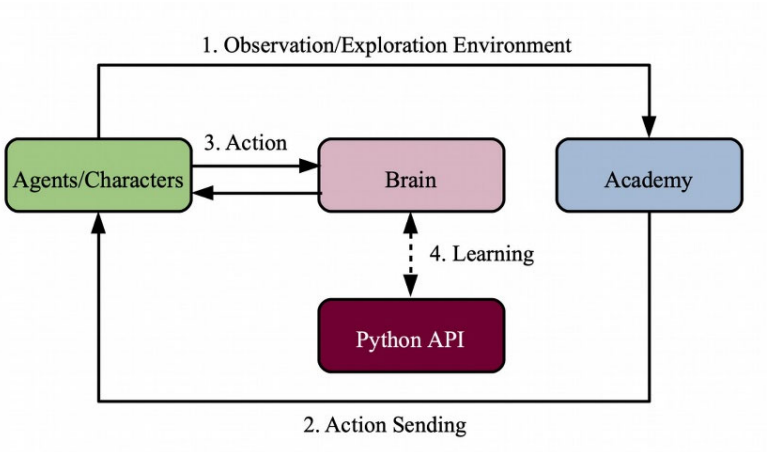

Figure 2 Component of ML-Agents

From figure 2, the Agents or characters contain observation function and action function. The Agents will reset an action when it found the Agent's steps property. Then, the Agents will start a new round of each movement; this is a machine learning in the dynamic environment. However, these ML-Agents have to define properties; in some cases, the agents occur a fear learning to an inaccurate goal learning in the games. Thus, there are curriculum learning; it is to determine lesson learn within the environment. Curriculum learning relates to curricula and lesson learning. The curricula are to choose for decision making of parameters in various situations. In case of the River Bridge environment, the wide of the bridge is varies. Lesson learning is associate the curricula; it is a sequence of learning from easy to difficult. Hence, the curriculum learning will help decreasing fear of decision-making learning for movement as displayed in figure 3.

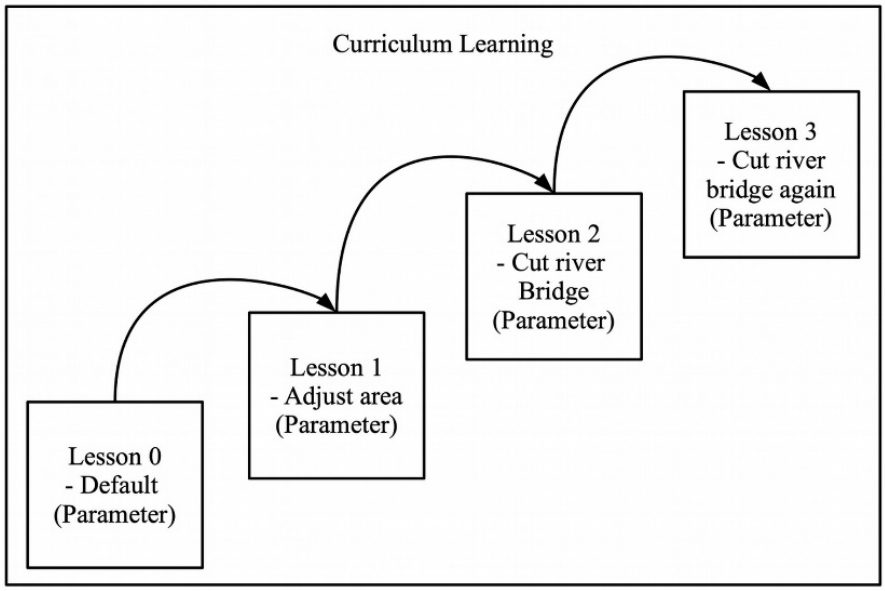

Figure 3 Curriculum Learning

For this reason, this research is to study and investigate curriculum learning for improving dynamic decision making for an enemy character in a tower defense game using Unity ML-Agents. The organization of the remainder of this paper is as follows. Section 2 is the objectives of the research. Then, material and methods are described in Section 3 while the results and discussion are explained in Section 4. Finally, the conclusion is described in Section 6.

2. Objectives

The objective of this study is to develop a dynamic decision making for an enemy character in a tower defense game using the Unity ML-Agents.

3. Materials and Methods

3.1 Materials

The research uses the Unity game engine version 2019.2 with the C# language for the determination of exploration and action of the Agents. Moreover, the Python language is applied with Unity ML-Agents, API and TensorFlow for the environment setup.

3.2 Methods

1) The first step of this research was to investigate and study the usability ML-Agents and reinforcement learning technique of Unity ML-Agents.

2) This step was to study the mechanism of the tower defense games using Unity ML-Agents and curriculum learning from a simple step to a complex step.

3) The next step was to design and develop a condition reward and punishment of character navigation and lesson learn of curriculum learning into an environment.

4) The fourth step was to create a script with the C# language of Agent to observe collection and action-taking. Next, agent led curriculum learning into the academy with the python language. Thus, brain is associated with agent and Academy for providing decision making for each action.

5) The final step was to test the performance of the model with Tensor Board, using environment (cumulative reward, episode length) and losses (value loss).

4. Results and Discussion

4.1 Results

Since this research is to develop an enemy character navigation in a tower defense game using reinforcement learning technique in Unity ML-Agents with curriculum learning. The agent is a problem solution for the enemy character movement in dynamic decision making. Curriculum learning composes of four lessons: lesson 0, lesson 1, lesson 2, and lesson 3. The scene environment in the research consists of ground, river, river bridge, target area, and restricted area. Thus, each lesson will be separated into levels for learning; lesson 0 will be defined to recognize the target area and the restricted area as presented in figure 4. Figure 5 is shown movement of the enemy characters and they will move to discover the areas when they found the target area that they will receive a reward.

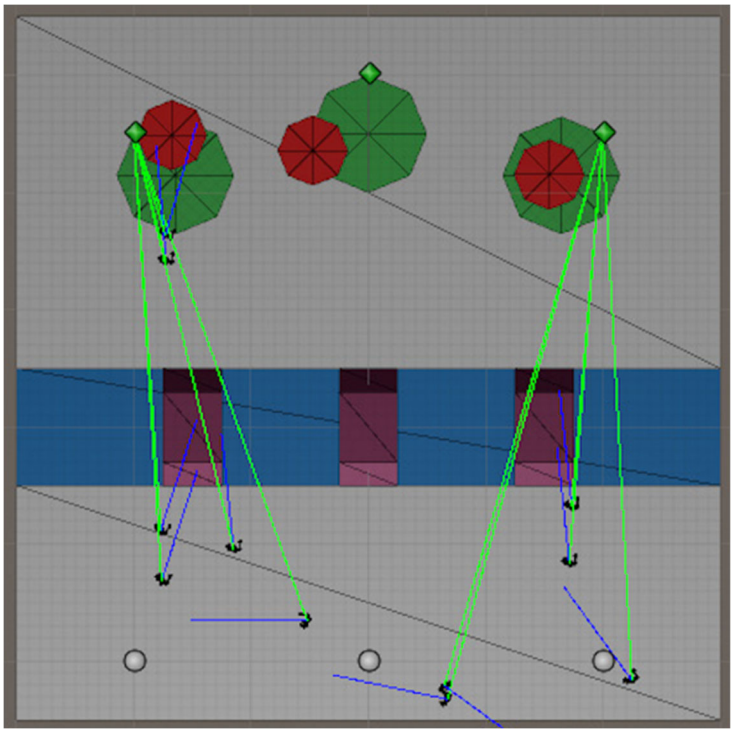

Figure 4 Top View of Enemy Characters

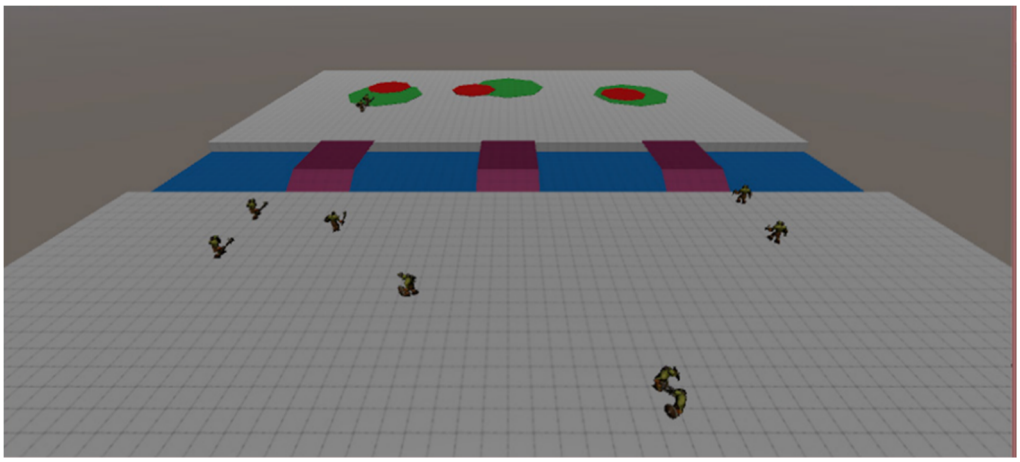

Figure 5 Movement of Enemy Characters

Next, lesson 1 is to learn for crossing the river bridge of the enemy characters. These enemy characters cannot cross the river bridge; they walk into the river in the first round of learning. The enemy characters will be punished by decreasing a reward. Thus, the characters will learn from the reward score, which they can be walked across the river bridge in the second round of lesson 1. Then, there are adjust the size of the river bridge from wide to narrow in lesson 2. As the same as lesson 1 results, the enemy characters cannot cross the river bridge because they don't recognize the new size of the bridge. Nevertheless, the enemy characters can move across the river bridge to the target area in the second round. The final lesson is lesson 3 that customizes the size of the river bridge smaller than the bridge in lesson 2. The enemy characters use more time to learn more than learning in lesson 2. However, the enemy characters can learn across the river bridge to the target area and avoid the restricted area as presented in figure 6.

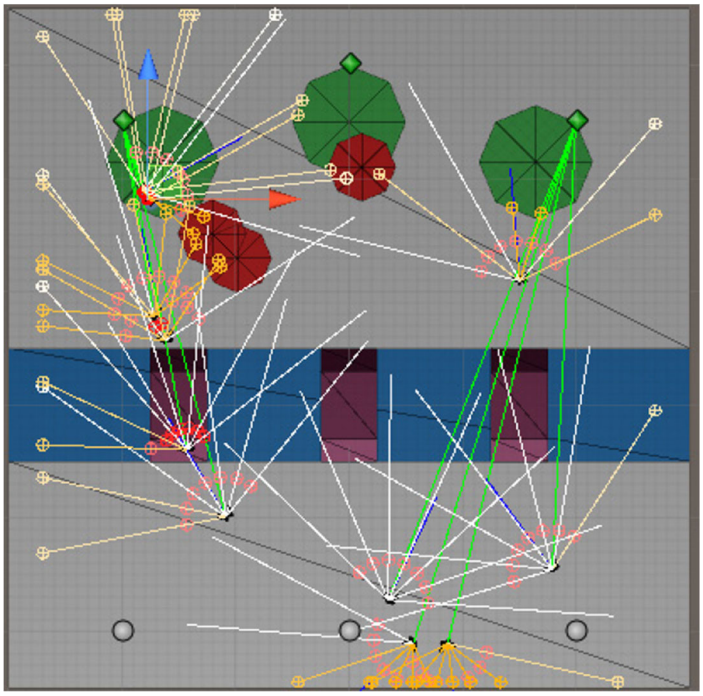

Figure 6 Enemy Character Movement of Lesson 3

In the curriculum learning, there are the identification of the steps of learning in each lesson, the experiment define maximum steps as 990,000 steps. Table 1 displays the average of each topic consists of steps, learning rate, cumulative reward, and episode length in the lesson 0, 1, 2, and 3. The lesson 0 uses 60,000 steps as the step of learning, and learning rate is 0.000865 second. The lesson 1 is defined the step of learning between 60,001 to 150,000 steps and it has learning rate as 0.000259 second. The lesson 2 is determined the step of learning between 150,001 to 300,000 steps and the lesson 3 is identified the step of learning between 300,001 to 990,000 steps. The learning rate of the last two lessons can measure as 0.000214 second and 0.000124 second respectively. The cumulative reward is reward points to accumulate for round of each lesson learn. From Table 1, the lesson 1 is the smallest cumulative reward because it is the first level to have increase learning. The first enemy characters cannot learn the new lesson in the first round, and they will be punished. However, in the second round of the first enemy characters, do not move on the same direction. These are the reasons that the cumulative reward is the lowest. On the other hand, the cumulative reward of the lesson 2 and the lesson 3 have gradually improved. Since, this experiment creates the curriculum learning of each lesson for learner as the enemy characters or the agents. Thus, there are many agents to learn in each lesson and each agent can learn many rounds until the best cumulative rewards. This is called episode length. As similar to the cumulative reward of the lesson 1, the lesson 1 is the highest episode length.

Moreover, figure 7 shows the cumulative rewards of all lessons. The lowest point of the graph is the average of the rewards in the lesson 1 that correspond to the value of the cumulative rewards from Table 1. For the same reason, the highest point of figure 8 is the average of the episode length in the lesson 1. That means the lesson 1 uses the maximum learning of the episode length.

Table 1 The Average Performance of Each Lesson

Lesson	Average				
	Steps	Learning Rate	Cumulative Reward	Episode Length	Value Loss
0	60,000	0.000865	0.671939	159.749848	0.099890
1	150,000	0.000259	-0.165550	468.056005	0.070165
2	300,000	0.000214	0.816457	254.561438	0.004538
3	990,000	0.000124	1.479682	335.675359	0.006491

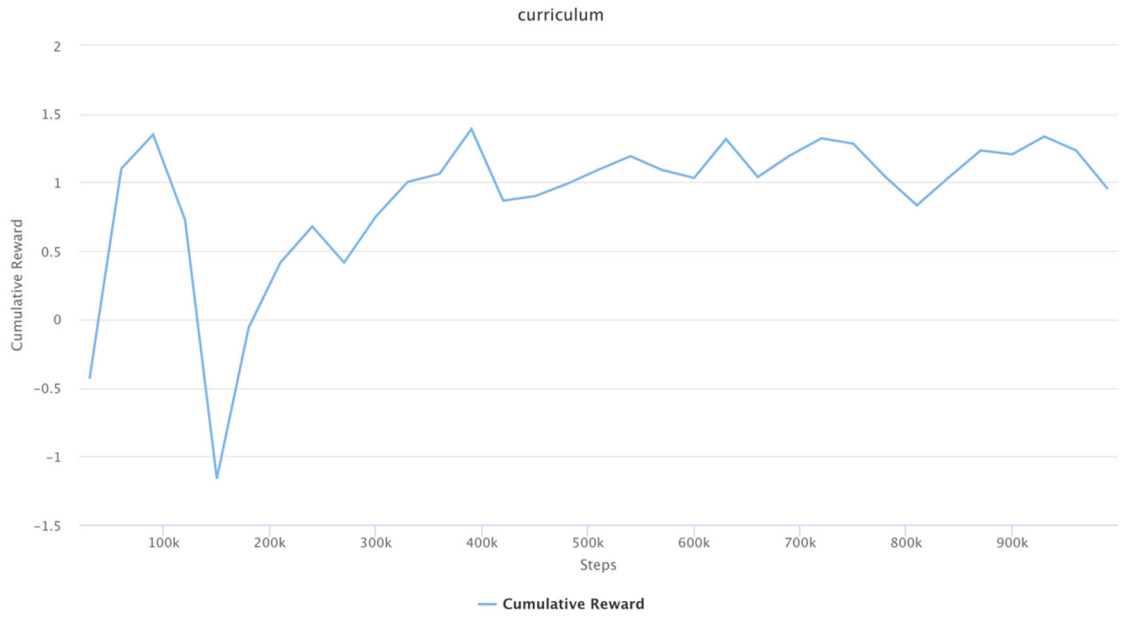

Figure 7 Cumulative Reward and Steps

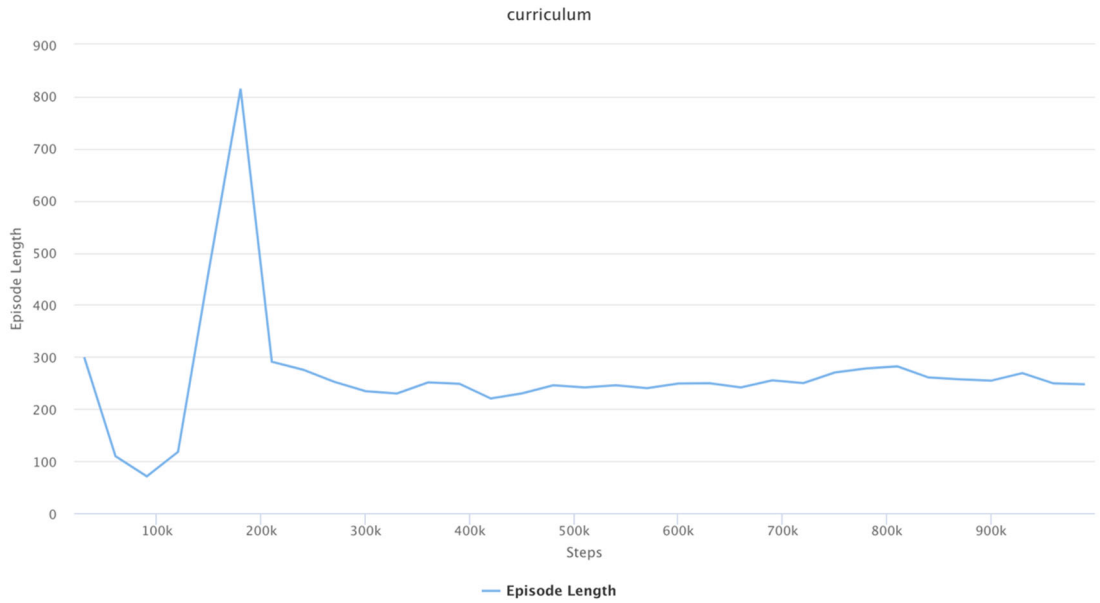

Figure 8 Episode Length and Steps

From Table 1, the value loss will increase while the enemy characters are learning. Then, the value loss will decrease when the learning is stable. The lesson 0 is the highest value loss because this is the first lesson learning of each agent. The value loss of the lesson 1 and the lesson 2 has gradually decreasing when there is a similar pattern of learning. Then, each agent has to learn in the new lesson as the lesson 3, the value loss has been increasing. Thus, figure 9 presents the value loss of each step.

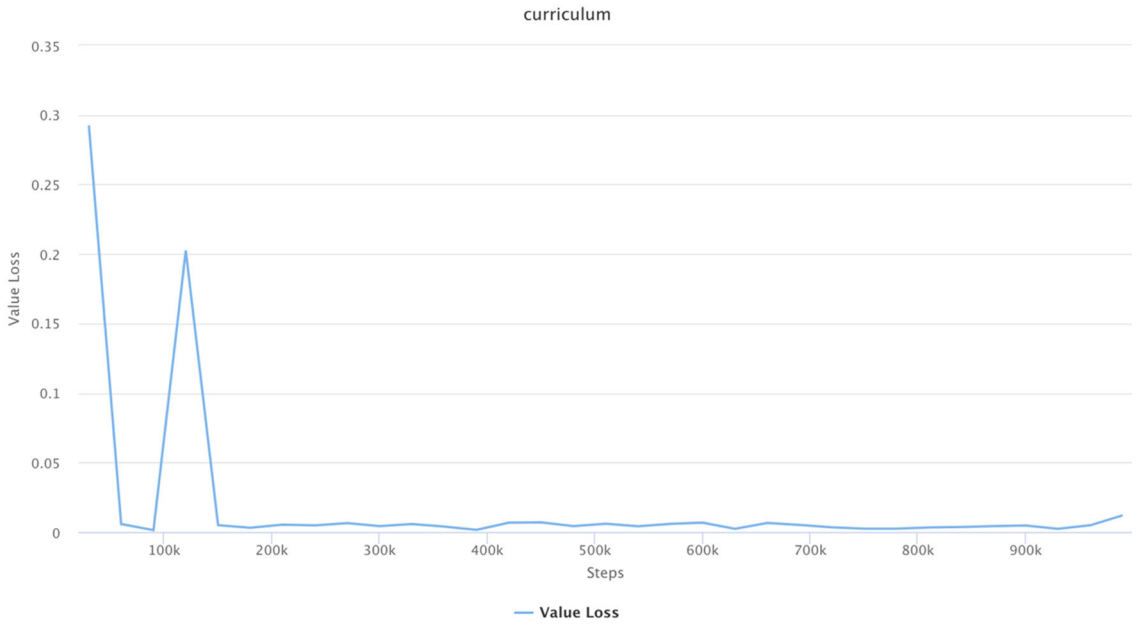

Figure 9 The Value Loss of Each Step

As the results, the learning with curriculum can identify the lesson learning for the enemy characters. Although, there are determine equivalent the learning rate both with curriculum and without curriculum, the cumulative reward with curriculum is better than without curriculum. The negative value of the reward means that the enemy characters cannot explore direction movement to the goal target. Moreover, the learning of the enemy characters without curriculum uses the episode length longer than the learning with curriculum. Thus, the value loss of the learning without curriculum is lower than the learning with curriculum, the enemy characters will move into the area of their characters. Therefore, Table 2 displays comparison of the learning with curriculum and without curriculum for max steps, cumulative reward, episode length, and value loss.

Table 2 Comparison of the Learning with curriculum and without curriculum

Environment/Loss	Learning	
	with Curriculum	without Curriculum
Max steps	990,000	990,000
Learning rate	0.000152	0.00152
Cumulative reward	0.891618	-1.592626
Episode length	263.662225	955.504450
Value loss	0.019373	0.013759

Moreover, figure 10 is compared to obtain the cumulative reward of the learning with curriculum and without curriculum. The blue line represents the cumulative reward with curriculum and the green line is for the cumulative reward without curriculum. Figure 11 is related to compare episode length with curriculum on the blue line and without curriculum on the green line. From two figures, the cumulative reward and the episode length with curriculum shown the learning better than the learning without curriculum.

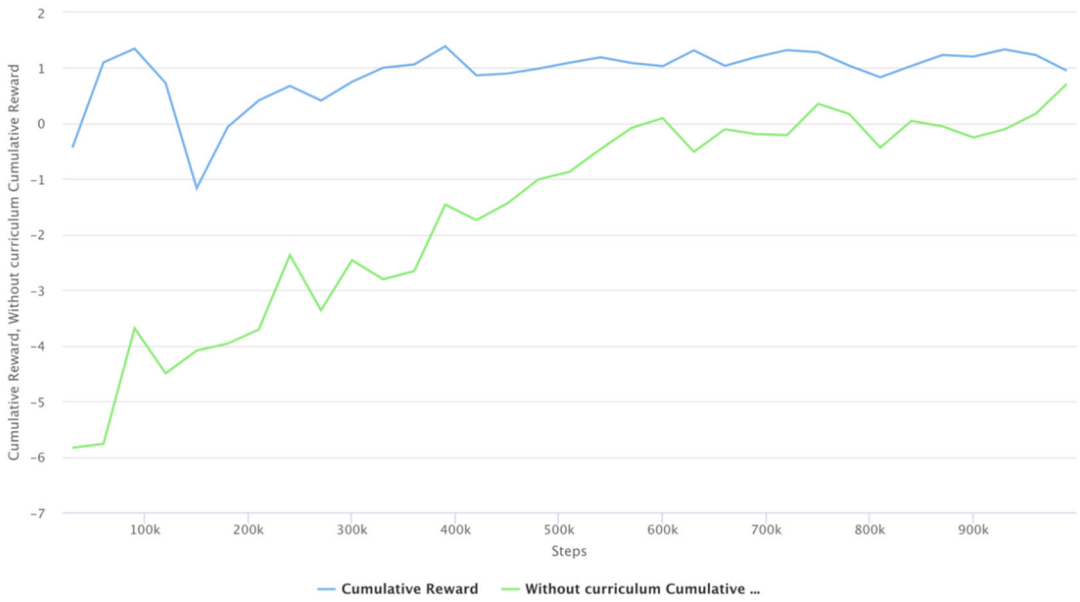

Figure 10 Comparison the Cumulative Reward with and without curriculum

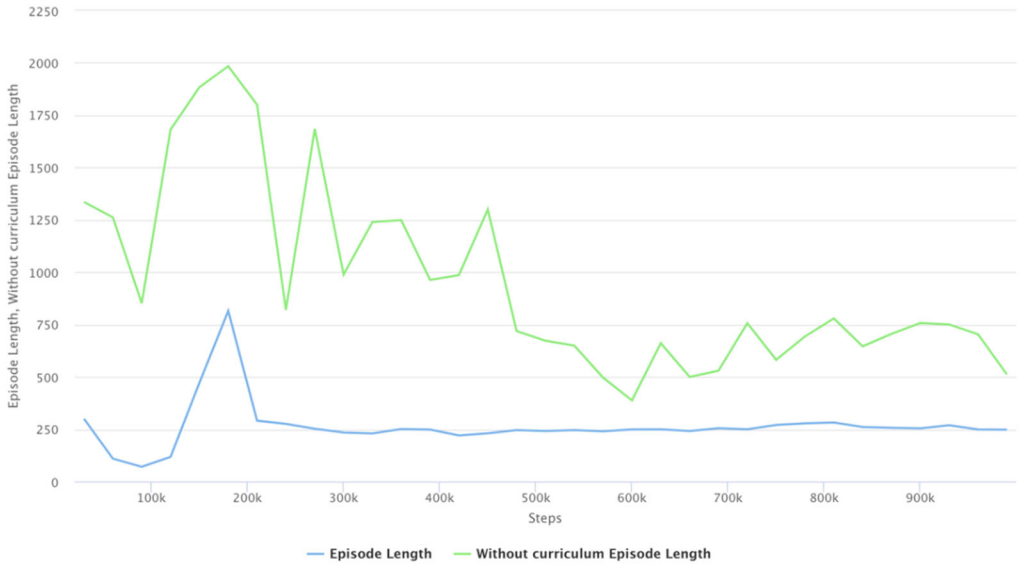

Figure 11 Comparison the Episode Length with and without curriculum

Furthermore, the hyperparameter process was conducted for the development dynamic decision making for the enemy characters. The hyperparameters in this experiment consist of beta, batch size, buffer size, hidden units, gamma, and time horizon. These parameters have an impact of the training process. Beta refers to controls the randomness of the policy, batch size and buffer size defines a number of experiences for the model such as state observations, actions, and rewards. Hidden units is connected neural network layer. Gamma determines the discount factor of the future rewards. Next, time horizon describes the amount of experiences for each agent. The hyperparameter values is chosen in this research as presented in Table 3.

Table 3 The hyperparameter values

Hyperparameter values	Values
Beta	1.0e-2
Batch size	128
Buffer size	2048
Hidden units	512
Gamma	0.99
Time horizon	128

4.2 Discussions

A tower defense game has the component of the environment and the objects, including a player and an enemy character. The players have to protect their towers and attack the enemies, the enemy's towers, and objects. An environment of the games has both a static environment and a dynamic environment. Generally, the enemy character will be defined direction movement to guess the direction in the static situation. The tower defense games are decreased an excitement while playing the games of the players. For these reasons, the direction movement of the enemy characters is developed to the ability for decision making of the dynamic environment using Reinforcement Learning (RL). This learning with curriculum is applied to develop decision making of the dynamic environment. After learning with curriculum, the enemy characters or the agents can make a decision to choose a path or direction for movement, such as avoid weapons from the players. They can make a decision better than the learning without curriculum by the cumulative reward. This solution can apply in several games such as strategy games, battle games, or turn-based games.

5. Conclusion

This research aims to develop dynamic decision making for an enemy character in a tower defense game using Unity ML-Agents. This study uses the curriculum learning for the enemy characters to comprise of lesson learning: lesson 0, lesson 1, lesson 2, and lesson 3. Therefore, there are definite direction movements of the enemy characters and the pattern of the right direction movement. Furthermore, the enemy characters, the agents, are implemented with the C# language via the Unity engine version 2019.2. Then, agent training is learned with reinforcement learning using a Python API to plugin with the Unity ML-Agents. As a result, the learning level from cumulative reward is improved direction movement of the agents continuously followed by lesson learning level. When agents can learn all the lessons, it can make a decision for the direction movement by itself.

6. References

Arther J., Vencent-Pierre B., Esh V., Yuan G., Hunter H., & Marwan M. (2018). Unity: A General Platform for Intelligent Agents. *ArXiv preprint arXiv: 1809.02627.*

Michailidis, L., Barcias, J. L., Charles, F., He, X., & Balaguer-Ballester, E. (2019, July). Combining Personality and Physiology to Investigate the Flow Experience in Virtual Reality Games. In *International Conference on Human-Computer Interaction* (pp. 45-52). Springer, Cham.

Philipa, A., Julian, T., & Elvis, A. (2014). Computational Intelligence and Tower Defence Games. *2011 IEEE Congress of Evolutionary Computation (CEC),* https://doi.org/10.1109/CEC.2011.5949738.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Gulec, U., Yilmaz, M., Isler, V., O'Connor, R. V., & Clarke, P. (2018, May). Adopting virtual reality as a medium for software development process education. In *Proceedings of the 2018 International Conference on Software and System Process (pp. 71-75). New York, United States*

He, Z., Shi, M., & Li, C. (2016, June). Research and application of path-finding algorithm based on unity 3D. In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS) (pp. 1-4). New York, United States*