Intelligent Avatar Interaction in 3D Computer Games

Miklos Balazs[1], Dorian Gorgan[2]
Technical University of Cluj-Napoca
Computer Science Department
Cluj-Napoca, Romania
[1]mklsbali@yahoo.com, [2]dorian.gorgan@cs.utcluj.ro

ABSTRACT
The artificial intelligence elements make the computer game more unpredictable and attractive by increasing the creative and emotional involvement of the players. An avatar can be directly controlled by the user through interaction techniques or at certain times, the avatar can make decisions alone in a behavior not controlled by the user. This paper attempts to determine experimentally how the intelligent characters may be trained, what are the level of intelligent behavior an avatar can reach through learning, or how the avatar makes decisions on his own through an adaptive behavior. The paper highlights solutions for efficient and successful training of the avatar.

Author Keywords
Intelligent agent, computer game, reinforcement learning, user interaction, avatar.

ACM Classification Keywords
H.5.2 User Interfaces.

DOI: 10.37789/rochi.2021.1.1.14

INTRODUCTION

Computer games have been developed a lot in recent decades. The introduction of artificial intelligence elements makes the game more interesting and unpredictable, which increases the creative and emotional involvement of players in the game evolution.

The characters in the game can adapt to unforeseen situations and can make their own decisions. The so-called NPC (Non-Player Characters) have been defined by Warpefelt [20] such as "characters within a computer game that are controlled by the computer, rather than the player". They can be represented for instance, by animals, monsters, vehicles, plant or various animated objects, and can play different roles. But not only NPC characters can be endowed with artificial intelligence, but even the avatar, as a representative of the user in the game.

There are not much research and many publications in the literature regarding the practical methodology of designing and developing computer games by introduction of artificial intelligence elements. This paper attempts to find out answers to the following questions:

- How to combine user-controlled interactive behavior with free and adaptive avatar behavior in a computer game?
- How does artificial intelligence work in games?
- How difficult is to design and train an intelligent agent and avatar for an adaptive behavior?
- What is the highest level an agent and an avatar can reach through learning?

For this purpose, the training of an agent is experienced in order to be able to pass alone to a destination, among several static or dynamic obstacles. Finally, the package of knowledge and skills accumulated by the agent is assigned to an avatar in a 3D game.

In an application or a game, the avatar can be controlled interactively and directly by the user or the avatar can have an adaptive, intelligent behavior, making movement decisions himself. The paper aims to explore experimentally such a complex interaction.

The paper is structured as follows. The next section presents some related works reported in the game literature. The following section describes the game as a framework for the experiments. Then, we present some conceptual solutions to develop an intelligent agent within an environment. The more consistent section experiment and validate solutions based on four use cases of an intelligent agent moving through obstacles. The following section validate the agent as a controlled and intelligent avatar within a game, and the last sections concludes the results of the research and the paper.

RELATED WORK

The scene crowded by NPC of various roles is developed in many games such as Dota 2 [4], Bioshock Infinite [8], Skyrim [10], or Mario Series [17].

Path planning algorithms based on artificial intelligence techniques are used in the game FEAR (First Encounter Assault Reconnaissance) [13]. The enemies talk about the actions and paths they follow. The psychological factors of players in perceiving the virtual world are very important.

The level of excitement and enjoyability of a user while playing games is given by the various events encountered during gameplay, by the behavior of the characters, or by the interaction with game objects and scene content elements [12]. There are different ways to generate content in the complex virtual world, such as procedural environment generation to different toolkits like DeepMind Control Suite [15], or OpenAI Gym [3]. Unity provides a toolkit called ML-Agents (Unity Machine Learning Agents) [6] that

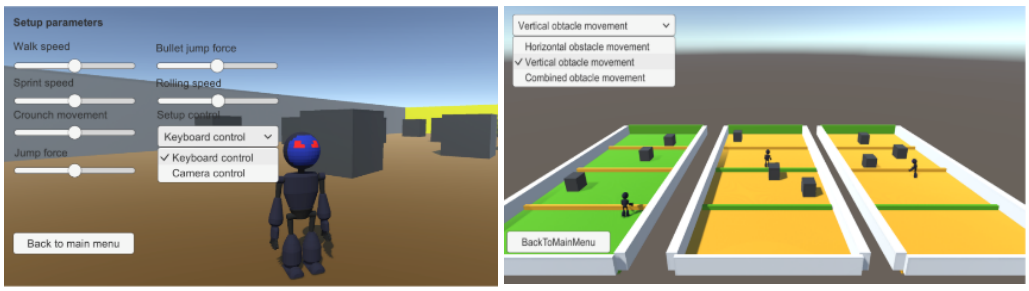

Figure 21. User interaction techniques in the game: (1) GUI by mouse and keyboard, (2) intelligent avatar behavior.

contains a framework for creating intelligent agents for computer games.

The neural networks are used to generate new frames of animation in order to provide more realistic movements to avatars that look similar to humans [5]. The avatars learn to walk, climb, or even jump over obstacles, while maintaining their balance.

Blaga et al., design a dodgeball non-player character (NPC) that can be integrated to give more dynamics to a digital game [2]. They explore the capabilities of the Unity ML-Agents technology and test its performance, while analyzing the ways in which it can be used to enrich a computer game.

Mario Kart [7] has used a Recurrent Neural Network for training to predict what controller inputs a player would use in a particular case, rather than concerning with a direct way to win. This is a method that works well for games that can be represented by a time series, such as a racing, and for agents that get as input visual information.

Belle et al. [1], by using behavior trees and Unreal Engine technology, explain how artificial intelligence can be created. They determine which behavior an agent should perform depending on its health amount - it should fight or run away.

GAME DESCRIPTION

This paper presents an application as a game, which concerns on experimenting solutions for user interaction techniques with intelligent avatars (Figure 21). An avatar can be directly controlled by the user through interaction techniques or at certain times, the avatar can make decisions alone in a behavior not directed by the user.

The avatars aim to walk toward a target position through a complex scene of moving obstacles. This paper attempts to determine experimentally how the intelligent characters may be trained, what are the level of intelligent behavior an avatar can reach through learning, or how the avatar makes decisions on his own through an adaptive behavior.

Through the graphical user interface, the user can set the value for some parameters of the movement such as walking speed, sprint speed, crouch movement, jump force, bullet jump force, and rolling speed.

The application, and the intelligent agents/avatar have been developed in the Unity technology and the ML-Agents toolkit. The scene of objects has been developed in Unity. The humanoid avatar that inherits the intelligent behavior of the agent, and its animations has been developed.

INTELLIGENT BEHAVIOR

The avatar and NPC objects should be able to make decisions according with information they get from environment. They need knowledge to understand the environment and be able to choose the next actions. The agent perceives the state of the environment and decides to perform an action that change the environment [19]. As a result of new state, the agent gets a reward. In order to complete convergently to the goal, the agent tries to maximize the reward (Figure 22).

One solution could be to describe the cases by behavioral patterns, and each pattern is described by rules, decision trees, or mathematically when you can describe conceptually and formally the cases. When this formalism is quite impossible for diversity or less understanding of the multiple cases, a solution is to describe the cases by examples, and particularly to train the objects by positive examples.

Such a solution is to train the objects for various cases in order to be able to understand a particular situation within the environment, to make a decision and to execute some actions. In fact, the behavioral patterns are embodied within models that are assigned to various objects within the game.

Reinforcement learning is a very promising solution through which the agents are trained to achieve a goal in an uncertain, potentially complex environment [14]. To get the machine to reach the appropriate target the agent gets either rewards or

Figure 22. The agent perceives the state of the environment and decides to perform an action in order to get a reward.

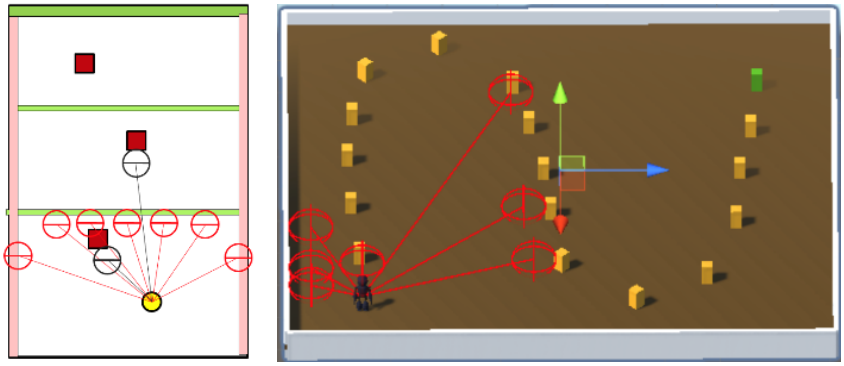

Figure 23. The agent senses the environment by casting rays around. In the left scene the agent senses the walls, the obstacles and the checkpoints. In the right scene the agent senses the next checkpoints on the trajectory, as well as the target position.

penalties for the actions it performs. Its goal is to maximize the total reward.

Although the designer sets the reward policy–that is, the rules of the game–he gives the model no hints or suggestions for how to solve the game [9]. The model learns how to complete the task and to maximize the reward by training, starting from totally random trials and finishing with sophisticated tactics and skills. This approach avoids completely the formal description.

The Unity ML-Agents technology provides the agent possibility to perceive the environment through rays casted around (Figure 23).

EXPERIMENTAL EVALUATION AND VALIDATION

This section analyses the results obtained during the training and the inference of the behavioral models. The TensorBoard visualization tool has been used to generate some graphics about different parameters of the training phase [16].

The trainings were done by the PPO (Proximal Policy Optimization) algorithms [11], [18]. The meaning of some parameters is the following:

• *Cumulative Reward* - represents the cumulative average reward of agents. This value should increase in the case of a successful training. Being a learning process, the value of the reward varies in small steps, depending on the bonuses and penalties received.

• *Episode Length* - the average length of episodes for agents. Represents the minimum number of steps by which the agent manages to complete his task, respectively to reach the destination

• *Value loss* - represents the value of the function that shows the loss in relation to the reward. Correlates to how well the agent is able to predict each step. It should have higher values at the beginning and during training and lower values at the end of training, when the reward has become stable.

• *Entropy* - shows how random are the decisions of the agent. The entropy value should decrease slowly during a successful training.

The avatar adaptive behavior has been experimented in four use cases of different level of difficulties: Demo, Level 1, Level 2, and Level3.

Demo basic use case

Before implementing more complex tasks, a demonstration of using the Unity ML Agents toolkit is implemented. In this basic demonstration, a game scene is implemented where the avatar and the target are two cubes. The avatar aims to reach the target in the scene. Each environment is limited by surrounding walls. If the agent cube meets a wall, its position is reset to the initial position and receives a penalty (negative reward). If the agent achieves the target it receives a positive reward. The experiment proved that the agent accumulates knowledge and becomes more and more smart in reaching the target.

Level 1 - The avatar avoids static obstacles

The first game scene (Level 1) is a training environment in which the agent must reach the target to avoid some static obstacles along the way. To speed up the learning process, we train the agent in parallel in several stages.

If the agent reaches a checkpoint, it receives a reward. If the agent collides with a wall or obstacle, it receives a penalty and the agent's position is reset to a random position at the beginning of the training environment, also the position of the obstacles changes according to a randomization logic. If the agent manages to reach the final target the episode ends, and it receives a reward. There is a time limit within which the agent must reach that destination. If it does not finish on time, the episode ends, and it receives a negative reward. The time required to complete each level was established heuristically, by direct manipulation of the avatar.

At this scene, 3 training phases were run in an environment where the agent had to reach a target, bypassing 3 obstacles

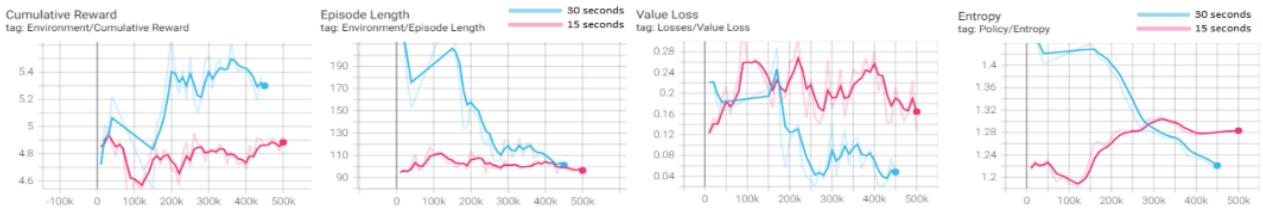

Figure 24. Level 1 – The agent is moving to the target through the static obstacles.

whose position changed randomly each time an episode ended.

1. In the first phase, a training from scratch was initiated, where the agent learned to walk the scene without obstacles.

2. The second training phase was initialized from the previously obtained model. Now the agent has learned to walk the stage with obstacles in 30 seconds.

3. The third training phase was initialized from the previous training phase model. In the end, the agent learned to walk the stage with obstacles in 15 seconds.

Below will be presented the results obtained in the trainings in phases 2 (30 seconds, blue line) and 3 (15 seconds, red line), (Figure 24). For a more efficient and fast learning, it is necessary the training hyperparameters (i.e., number of hidden layers, number of units or sets of nodes in the hidden layer, learning rate, number of iterations, optimizer, activation function, batch size, etc.), are configured as well as possible.

The second training phase (30 seconds) needs 450,000 steps, and the third training needs 500,000 of steps. The maximum reward (Cumulative Reward), obtained in the case of 30 seconds, reaches approximately the value of 5.4. It can be seen that the reward drops to 4.8 when the agent has only 15 seconds to go through the scene.

It can be seen that the length of the episodes decreases from the value of 216 to the value of 96, especially in the case of complete learning of the scene in 30 seconds (blue line). In the case of training in phase 3, the length of the episode becomes stable with a value of 95-100 (red line). This means that the agent has a positive evolution. At the beginning it needs a sequence of 200 steps for the complete passing through the scene, and finally it manages to complete the scene in 95-100 steps.

The loss decreases from 0.28 to 0.04 in the case of completing the scene in 30 seconds (blue line). This means the agent has a good evolution. If the agent has only 15 seconds, an increase in loss can be observed (red line). In this case, the agent has more difficulty completing the scene, when it has less time available.

The behavior of the entropy function is similar to that of the loss function. If it has 30 seconds available, the entropy decreases from 1.4 to 1.22, meaning the agent makes fewer random decisions, so it evolves. A small increase of entropy (up to 1.28) in the case of 15 seconds, means that the agent has more difficulties, having less time.

These graphs highlight the progress of the agent's learning in completing this game scene.

Level 2 - The avatar follows a given trajectory

In this case, the agent's goal is to follow a path marked with checkpoints. They must be followed in order and the agent is not allowed to bypass a checkpoint. The agent must not start at every failure from the starting position but must start from the last checkpoint where it arrived.

We use assisted learning by introducing additional constraints in the form of interior walls. This assistance was necessary because without the interior walls, the agent failed to learn the curved trajectory correctly:

1. The first phase a training process was run where the agent learned to complete the scene with the inner helper walls in 50 seconds.

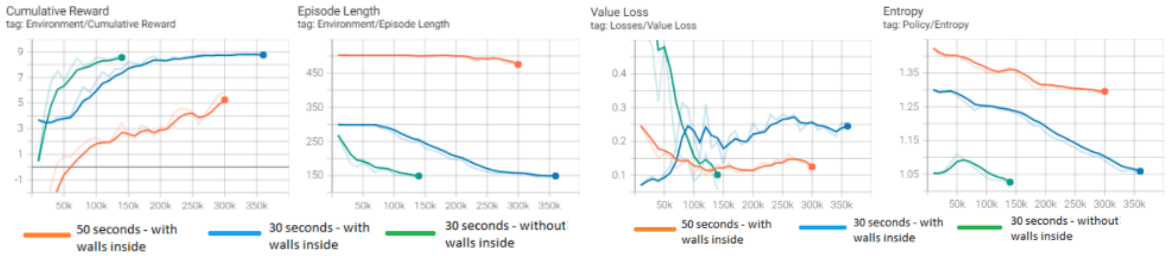

Figure 25. Level 2 – Cumulative Reward, Episode Length, Value Loss and Entropy graphical presentation for movement of the agent along a given trajectory.

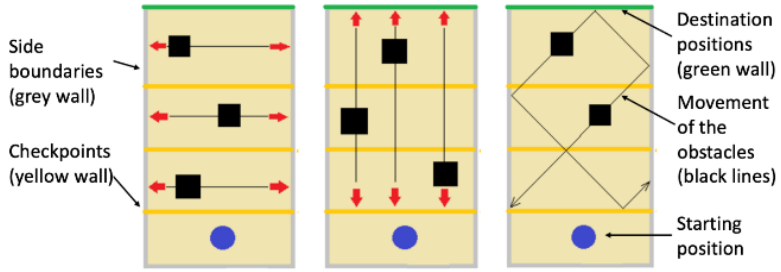

Figure 26. Level 3 – Obstacles are moving horizontally (left), vertically (middle) and on both directions (right).

2. After that, another training process was initiated from the first model where the agent learned to complete the scene in 30 seconds.

3. Finally, another training process from model 2 was initiated, where the agent learned to complete the scene in 30 seconds, but without auxiliary walls inside the training environment.

As a first observation, the length of the curves on the graphs differs from the three training processes (Figure 25):

1. The first training (orange), in which the agent learned to complete the scene in 50 seconds, has approximately 260,000 steps. The maximum reward is about 5.8.

2. The second training (blue), in which the agent learned to complete the scene in 30 seconds has about 350,000 steps. The reward increases to a value of 8.7.

3. The third training (green) has the fewest steps, 130,000) and a reward almost as high as that obtained after the second training. This means that the agent learned in a shorter time to complete the scene in 30 seconds without walls / obstacles inside. Obviously, each stage / training was initialized with the previous training. Otherwise, such effective learning would not be possible.

The episode of the Episode Length function highlights the good evolution of the agent's learning. It can be seen that the average length of the episodes decreases with the training described above.

1. At the first training (orange) the shortest episode length is about 475 at step 300,000.

2. In the second training (blue), which was initialized from the first training, the shortest episode length is 148 at step 350,000.

3. After the third drive (green), which was initialized from the second drive, the shortest episode length is 144.

In the graph of the loss function it can be seen that in most trainings (orange and green) the loss decreases. This means a good evolution of the agent. When training to complete the scene in 30 seconds with interior walls (blue), unlike the two training, it is observed that the loss increases, but still, it is not very high. The biggest evolution of the agent, in terms of losses, is the third training (30 seconds without interior walls). In this case, a sudden decrease in the values of the loss function can be observed from a value of over 0.4 to a value below 0.1.

The entropy function has a decrease in the 3 training phases. This means that the agent makes fewer random decisions, so he evolves in learning to complete the scene. It can be seen that the entropy decreases from the first training phase to the third training phase. The best entropy, 144 was obtained after the third training. The graphs show good results, so our agent evolved in this case study as well.

Level 3 - The avatar avoids dynamic obstacles

In this game scene the obstacles are dynamic. This means the obstacles follow a certain trajectory during the game according to a certain algorithm. The goal of our agent is to cross the stage successfully without encountering obstacles. The training environment with checkpoints, walls and the final target is similar to the training environment at the Level1, the difference lies in the behavior of the obstacles.

Figure 27. Level 3 – Cumulative Reward, Episode Length, Value Loss and Entropy graphical presentation for avoiding the obstacles with horizontal movement.

The agent has 15 predefined seconds to complete this level of play.

Obstacles can have 3 types of behavior: horizontal movement, vertical movement or combined movement (both horizontal and vertical), (Figure 26).

Horizontal movement of obstacles

This scene contains 3 obstacles, which have a movement in the horizontal direction (x-axis). We used two configurations of hyperparameters, and in both configurations the agent was trained in 2 million steps, for the scene is more complex. In the second configuration the parameters batch size, buffer size and num_epoch have higher values than the first configuration.

The reward function shows that in both hyperparameter configurations, the agent's reward increases rapidly between steps 200,000 and 300,000 (Figure 27). At step 300,000 the reward has a value of 2.2 for configuration 1, and a value of 1.9 for configuration 2. Somewhere in step 700,000 the value of the reward in configuration 2 exceeds the value of the reward in configuration 1. At step 2,000,000 the rewards become stable in both configurations, reaching an approximate value of 2.7.

The length of the episodes decreases sharply between steps 200,000 and 300,000. The length it reaches from a value of over 200 to a value of about 100 after the end of the training, which means that the agent is able to complete the scene in a few steps.

The graph of the loss function shows that the training by the hyperparameters of configuration 1 has a lower loss than the one with configuration 2. Both trainings are successfully, because in both cases the loss decreases.

The entropy graph similarly shows the good evolution of the agent, because during the training a decreasing entropy can be observed, which means that the agent makes fewer random decisions. The entopy function of the training in configuration 1 has a lower value than the entropy in configuration 2.

Each graph highlights the positive evolution of the agent's training. Configuration 1 is better for loss and entropy functions, and configuration 2 is better for functions of reward and length of episodes.

Vertical movement of obstacles

This scene contains 3 obstacles moving vertically, along the direction of z axis. The agent has 15 seconds to reach the destination, bypassing obstacles.

In this experiment we analyze the trainings, by using the two previous configurations. The training is divided into 2 major phases, the total length of the training is 1 million steps (Figure 28):

1. The first training phase contains 500,000 steps, and it is initiated from the neural network model that knows how to walk the scene without obstacles (steps 0 - 500,000).

2. The second training phase is initiated from the first training phase, and it contains a total of 500,000 steps (steps 500,000 - 1,000,000).

Between steps 0 - 500,000 the reward function has a greater variation. Training with configuration 1 (green line), at step 500,000 has a higher value (2.4) than training at configuration 2 (1.9), (grey line). The length of the episodes shows a decrease from 200 to about 120 in both configurations.

At steps 500,000 - 1,000,000 it can be seen that the reward has reached in configuration 1 (orange line) up to the value of 3, as opposed to configuration 2 (blue line) which has the maximum reward 2.6. The length of the episodes decreased

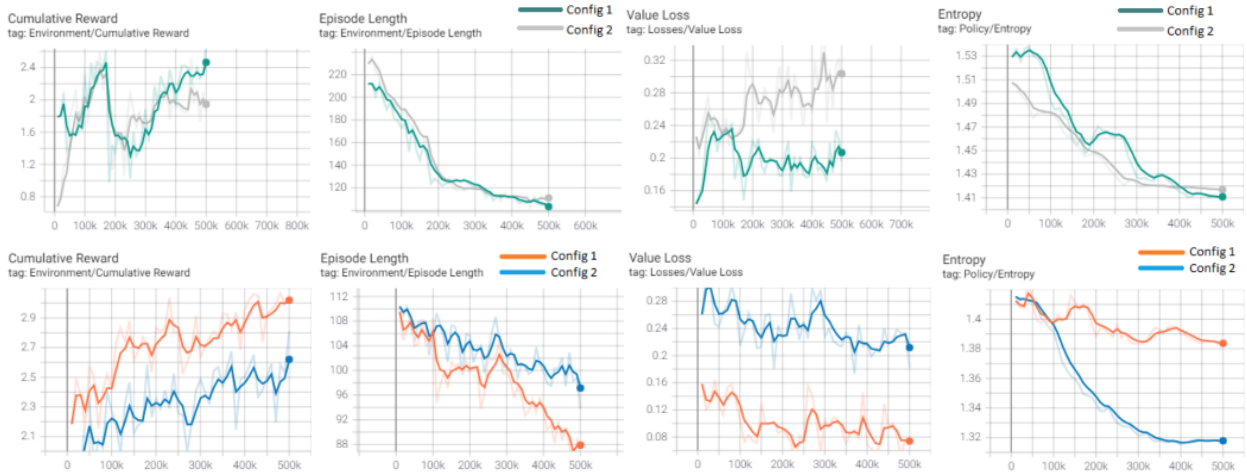

Figure 28. Level 3 – Cumulative Reward, Episode Length, Value Loss and Entropy graphical presentation for avoiding the obstacles with vertical movement. Steps: 0 - 500,000 upper diagrams (green line for Configuration 1, and grey line for Configuration 2), steps: 500,000 - 1,000,000 bottom diagrams (orange line for Configuration 1, and blue line for Configuration 2).

Figure 29. Level 3 – Cumulative Reward, Episode Length, Value Loss and Entropy graphical presentation for avoiding the obstacles with combined movement. There are 3 phases: steps 0 - 500,000 (blue line), steps 500,000 - 1,000,000 (red line), and steps 1,000,000 – 1,500,000 (green line). Blue, red and green lines are in a sequence of 500,000 steps each phase.

from 110 to 88 in the case of configuration 1 and up to 100 in the case of configuration 2. In conclusion, after 1 million steps the agent has an increase in reward from a value less than 1, up to a maximum value of 3. The length of the episodes decreased from 220 to 88.

It is obvious that the training graphs with configuration 1 show better values than with configuration 2 both in terms of reward and length of the episodes. Both the increase in the reward and the decrease in the length of the episodes show the good evolution of the agent in both training cases.

In steps 0 - 500,000 the loss functions have a higher oscillation in both training configurations, because the reward has not yet stabilized. Entropy, in both cases has a decrease and after step 500,000 reaches an approximate value of 1.4.

At steps 500,000 - 1,000,000 the loss function begins to decrease, which is a good evolution that the reward has stabilized. The entropy function of the training by the configuration 2 has a larger decrease than the other curve, reaching the value of 1.32.

In conclusion, the lowest loss (0.08) was obtained in the case of training with configuration 1, and the best entropy was obtained in the case of training with configuration 2 (1.32). These data show that the agent evolved well and managed to successfully learn the task throughout the training. In terms of reward, episode length and losses, the configuration 1 of the hyperparameters has achieved better results than configuration 2.

Combined movement of obstacles

In this scene there are only 2 obstacles in order to reduce the training time and required computation resources. The obstacles have combined movements along the x-axis and along the z-axis. The agent has only 15 seconds to reach the destination.

The training has 3 phases that are performed in 1,500,000 steps totally (Figure 29):

1. The first training phase (steps 0 - 500,000) is initialized with the neural network model that is the result of learning to walk the scene without obstacles.

2. The second training phase (steps 500,000 - 1,000,000) is initialized with the final result of the first phase.

3. The third training phase (steps 1,000,000 - 1,500,000) is initialized with the final result of the second phase.

At each training phase, configuration 2 is used, rather than configuration 1, because it has higher values for the hyperparameters batch_size, buffer_size and num_epoch.

Each reward function has an evolution compared to the previous phase. After 500,000 steps, the reward reaches approximately 2. After 1 million steps the reward increases to almost 2.4, and after 1.5 million steps the maximum reward value is approximately 2.6. The largest increase in reward function is in the first phase, between steps 0 - 500,000. This means that the agent learned the most in the first phase of training.

Therefore, the value of the rewards increased from 0 to 2.6 after the 3 training processes. This means a good evolution of the agent training process.

The functions of the episode lengths have a decrease in the third training processes. The average length of the episodes decreases from a value higher than 140 to a value lower than 100, which shows the good evolution of the agent.

The graphs of the loss functions have many oscillations at the third training phase. The oscillation becomes higher after the third training phase, but the loss is smaller. These oscillations of the loss function show the higher difficulty of this scene compared to the others.

The entropy graph shows a good result, because the entropy decreases by increasing the number of training steps. After the first training phase, the entropy has the best value of 1.4. After the second training phase the value of the entropy reaches 1.3, and finally after the 1.5 million steps, the entropy reaches almost 1. So, the agent no longer makes so many random decisions, which shows a good evolution of the learning process.

In conclusion, each graph shows a good development of the agent, in the case of the scene that contains dynamic obstacles with combined movements.

INTELLIGENT AVATAR WITHIN THE GAME

The behavior of the intelligent agent within the scene may be analyzed through the game and various scenes of objects. The two interaction techniques are exemplified: (1) the avatar is directly controlled by the user through interaction techniques, and (2) at certain times, the avatar can make decisions alone in a behavior not directed by the user.

In the case (2) there are four scenes of different level of difficulties: Demo, Level 1, Level 2, and Level3. These scenes highlight intelligent behavior and evolution of the avatar, as well as the limits of its adaptive ability. The avatar goes through the scenes successfully, but sometimes makes mistakes determined by the speed of the objects and the complexity of the scene.

CONCLUSIONS

In this paper we present some approach of training agents through four use cases in order to assign the accumulated acknowledge to the avatar in the game. The experiments have proved successfully the efficient training and the adaptive behavior of the avatar through the dynamical obstacles in the scene.

First of all, machine learning is a complex concept, difficult to understand. It is very difficult and time consuming for a developer to become an expert in this field. For instance, it is difficult to determine whether all the learning processes the agent goes through are well configured, or environment parameters are well configured, or the learning environments are well defined. It is certain that there are many ways to improve the solutions.

Another issue that has been identified is the computing time that is closely related to the performant hardware resources (e.g. cloud or cluster with virtualized GPUs) the performance would be significantly improved.

Future works will focus on extending the intelligent behavior to more various cases by composing hierarchically the complex behavior from basic abilities.

REFERENCES

[1] Belle, S., Gittens, C. and Graham, T.N., Programming with Affect: How Behaviour Trees and a Lightweight Cognitive Architecture Enable the Development of Non-Player Characters with Emotions. In 2019 IEEE Games, Entertainment, Media Conference (GEM) (pp. 1-8). 2019.

[2] Blaga B.C.Z., Gorgan D., Performance Analysis in Implementation of a Dodgeball Agent for Video Games. International Journal of User-System Interaction 12(4), pp.225-240, (2019).

[3] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J. and Zaremba, W., Openai gym. arXiv preprint arXiv:1606.01540. 2016.

[4] Drachen, A., Yancey, M., Maguire, J., Chu, D., Wang, I.Y., Mahlmann, T., Schubert, M. and Klabajan, D., Skill-based differences in spatio-temporal team behaviour in defence of the ancients 2 (dota 2). In 2014 IEEE Games Media Entertainment, 2014.

[5] Holden, D., Komura, T. and Saito, J., Phase-functioned neural networks for character control. ACM Transactions on Graphics. 2017.

[6] Johansen, M., Pichlmair, M. and Risi, S., Video Game Description Language Environment for Unity Machine Learning Agents. In 2019 IEEE Conference on Games (CoG) (pp. 1-8). 2019.

[7] Lei, J., Chen, S. and Zheng, M., Using Machine Learning to Play the Game Super Mario Kart. 2019.

[8] Lizardi, R., BioShock: Complex and alternate histories. Game Studies, 2014.

[9] Osinski B., Budek K., What is reinforcement learning? The complete guide, 2018, https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/

[10] Puente, H. and Tosca, S., The Social Dimension of Collective Storytelling in Skyrim. In DiGRA Conference. 2013.

[11] Schulman J., Wolski F., Dhariwal P., Radford A., Klimov O., Proximal Policy Optimization Algorithms, arXiv:1707.06347v2, 2017.

[12] Shaker, N., Yannakakis, G. and Togelius, J., Towards automatic personalized content generation for platform games. In Sixth Artificial Intelligence and Interactive Digital Entertainment Conference. 2010.

[13] Spittle, S. Did This Game Scare You? Because it Sure as Hell Scared Me! FEAR, the Abject and the Uncanny. Games and Culture, 6(4), pp.312-326. 2011.

[14] Sutton R.S., Barto A.G., Reinforcement Learning: An Introduction, Second Edition, MIT Press, Cambridge, MA, 2018.

[15] Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D.D.L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A. and Lillicrap, T., Deepmind control suite. arXiv preprint arXiv:1801.00690. 2018.

[16] TensorBoard as TensorFlow's visualisation toolkit, https://www.tensorflow.org/tensorboard.

[17] Togelius, J., Karakovskiy, S., Koutn k, J. and Schmidhuber, J., Super Mario evolution. In 2009 IEEE symposium on computational intelligence and games, pp. 156-161, 2009.

[18] Understanding PPO Plots in TensorBoard https://medium.com/aureliantactics/understanding-ppo-plots-in-tensorboardcbc3199b9ba2.

[19] Unity Machine Learning Agents Toolkit, https://github.com/Unity-Technologies/ml-agents.

[20] Warpefelt, H., The Non-Player Character: Exploring the believability of NPC presentation and behavior. Doctoral dissertation, Department of Computer and Systems Sciences, Stockholm University. 2016.