© Journal of the Technical University - Sofia
Plovdiv branch, Bulgaria
"Fundamental Sciences and Applications"
Vol. 26, 2020

Applying Unity as a simulation platform for modelling the behaviour of a Non-Player Character and for illustrating the work of studied algorithms

Dilyana Budakova, Velyo Vasilev

Technical University of Sofia, Branch Plovdiv, 61, "Sankt Peterburg" blvd. Bulgaria

dilyana_budakova@yahoo.com, velyo.vasilev@abv.bg

Abstract. Unity is a universal simulation platform that consists of a game engine and a graphical user interface. Unity is now used by a large community of game developers, researchers, teachers, and students. In this article Unity simulation platform is used for e modelling Non-Player Character (NPC) behaviour and for illustrating the work of algorithms, which are studied by students. It is expected that this will increase the interest of the students to create NPCs, which perceive their surroundings, make decisions and take actions; to study algorithms and to carry out researches.

1. Introduction

Unity is a universal simulation platform that consists of a game engine and a graphical user interface. Unity engine is an ideal simulation platform. Unlike many of the research platforms like DeepMind Lab [1], Arcade Learning Environment (ALE) [2], Project Malmo [3], VizDoom [4], the Unity game engine is a flexible and it is not restricted to any specific genre of gameplay or simulation [5]. The Unity Editor enables rapid prototyping and development of games and simulation environments.

Unity is now used by a large community of game developers, researchers, teachers, and students. It is possible to make everything from simple gridworld problems, small mobile and browser-based games to complex strategy games, physics-based puzzles, high-budget console games, AR/VR experiences or multi-agent competitive games.

Games have long been a popular area of Artificial Intellect (AI) research, and have become a fast growing software industry since 1990s. Main areas where AI applies to games are: Non-Player Character AI; Decision Making and Control; Machine Learning; Interactive Storytelling; Cooperative Behaviours, Player Control, Content Creation [6][7].

In this article Unity simulation platform is used for modelling the behaviour of an NPC and illustrating the work of algorithms, which are studied by the students.

Various behaviours such as patrolling; enemy evasion; circumventing static or dynamic obstacles have been realized. The work of one of the most commonly used pathfinding algorithms A*(pronounced "A star") [8][9] has been illustrated. The A* algorithm is precisely what the Unity navigation system uses [10].

 Online ISSN 2603-459X

It is expected, that the visualization of the algorithms by using 3D projects and Virtual/Mixed reality [12] will increase the interest of the students to the studied algorithms, to carry out researches, as well as to create NPCs, which perceive their surroundings, make decisions and take actions.

2. Illustrating the A* algorithm

In order to illustrate how the A* pathfinding algorithm works, a 3D scene and an NPC have been realized. The user selects the destination, which the NPC has to reach. The simulation environment of Unity - Visual Studio.NET has been used and the programming language is C#. Fig.1 shows an example maze, along with a constructed grid and marks on the openings, which can be passed through. The table, with the necessary data for the algorithm to work, is shown on fig.2. On fig. 3 and fig. 4 the maze is represented by a weighted graph and the order of traversing the nodes from the starting node A to the target node J has been noted.

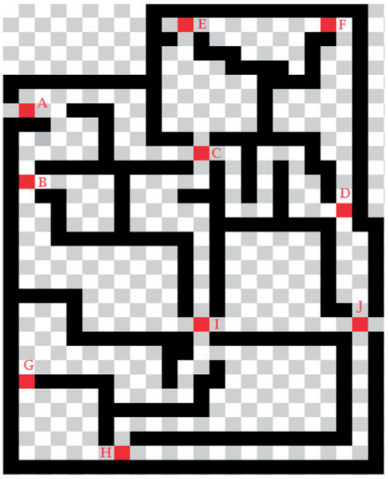

Figure 1. A maze with a constructed grid and marked passable openings

Node	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous node
A √	0	35	35	-
B √	9	31	40	A
C √	16	22	38	A
D	37	9	46	C
E	26	32	58	C
F		23		
G	49	25	74	I
H	48	24	72	I
I √	30	10	40	B
J	40	0	40	I

Figure 2. Table of the data, necessary for the work of the A* algorithm

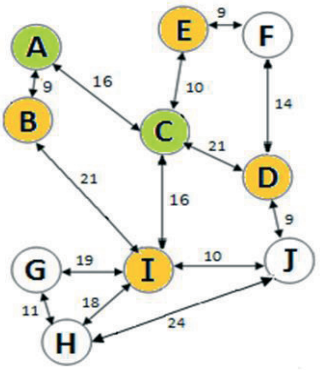

Figure 3. Representation of the maze by a weighted graph. A path from node A to node J has to be found. A path from node A to node C is suggested.

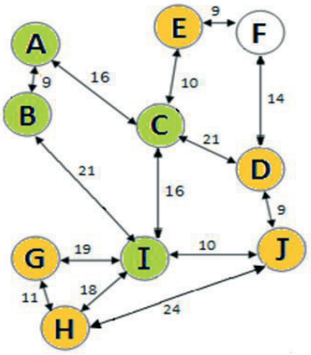

Figure 4. After further calculations the suggested path is from node A to node B, after which node I and then to the goal node J.

3. Implementing the behaviour of the NPC.

In order to implement NPC behaviours such as patrolling; path following; enemy evasion and circumventing static or dynamic obstacles, The Unity navigation system is used.

The Unity navigation system allows you to create characters that can intelligently move around the game world, using navigation meshes that are created automatically from your Scene geometry. Dynamic obstacles allow you to alter the navigation of the characters at runtime, while off-mesh links let you build specific actions like opening doors or jumping down from a ledge [11].

NavMesh (short for Navigation Mesh) is a data structure which describes the walkable surfaces of the game world and allows to find path from one walkable location to another in the game world. The data structure is built, or baked, automatically from your level geometry [11].

NavMesh Agent component help you to create characters which avoid each other while moving towards their goal. Agents reason about the game world using the NavMesh and they know how to avoid each other as well as moving obstacles.

NavMesh Obstacle component allows you to describe moving obstacles the agents should avoid while navigating the world. A barrel or a crate controlled by the physics system is a good example of an obstacle. While the obstacle is moving the agents do their best to avoid it, but once the obstacle becomes stationary it will carve a hole in the navmesh so that the agents can change their paths to steer around it, or if the stationary obstacle is blocking the path way, the agents can find a different route [11].

Fig. 5 shows the realization of a 3D scene for demonstrating the behaviour of the Non-Player Character (NPC). Pathfinding addresses the problem of finding a good path from the starting point to the goal, avoiding obstacles, avoiding enemies, and minimizing costs in the game. Movement addresses the problem of taking a path and moving along it. The gaming industry has found out that the best results are achieved by using both pathfinding and movement algorithms [6][8][9].

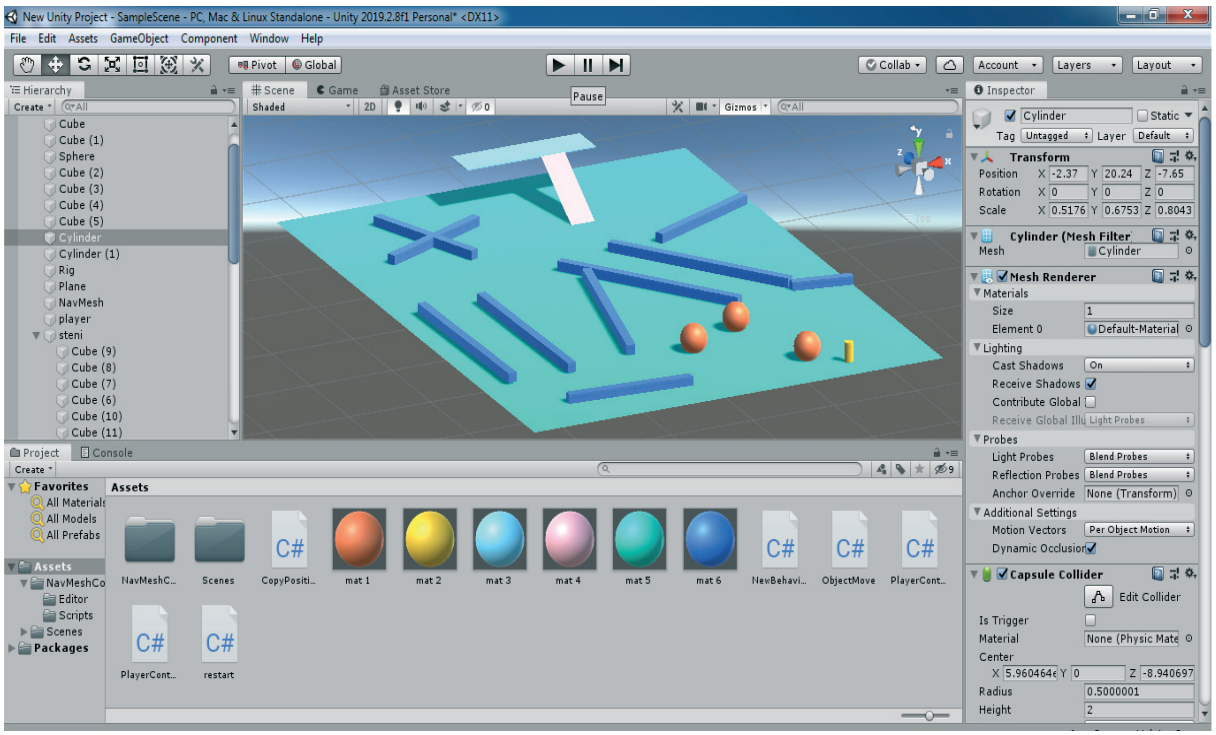

Figure 5. 3D scene necessary to demonstrate the behaviour of the NPC.

An NPC is usually a character in the game, a vehicle, a robot, a group of creatures, a country, or a civilization. The agent has to observe the surrounding environment, make decisions based on the

observations and take actions in the environment. That is to say a loop of Sensing, Thinking and Acting has to take place. At the sensing stage the agent detects or is given information about the surrounding environment, which affects its behaviour (for example: nearby threats, collectible items or points). At the thinking stage the agent makes a decision in response to the received information (for instance, it assesses whether it is safe enough to collect points or items, whether it should attack or hide first). During the action stage the agent performs actions, based on the decision taken (such as beginning to move towards an object). After completing the loop, the circumstances have changed due to the actions of the characters, and the loop must be repeated with new data.

When the abilities of the agent to avoid obstacles, patrol, evade an enemy or follow a certain path have been realized, then it is possible for finite-state machines to be implemented. They are abstract machines with finite memory and a limited number of internal states, which have wide use in games with NPCs. The internal states could be: patrolling, attacking, retreating, idling, searching, etc. The rules of the NPC for changing a state can, for example, be the following: „If the patrolling NPC sees an enemy, then it has to attack it"; „If the NPC no longer sees the enemy, then it goes back to the state of patrolling."

4. Conclusion

In this article Unity simulation platform is used for modelling the behaviour of a Non-Player Character and for illustrating the work of algorithms, which are studied by the students Various behaviours such as patrolling; enemy evasion and circumventing static or dynamic obstacles have been realized. The work of one of the most commonly used pathfinding algorithms has been illustrated - namely A*(pronounced "A star") [8][9]. The A* algorithm is precisely what the Unity navigation system uses[10]. Unity is a universal simulation platform that consists of a game engine and a graphical user interface. Unity is now used by a large community of game developers, researchers, teachers, and students. It is expected, that the visualisation of the algorithms by using 3D projects and Virtual reality will pique the interest of students to learn the algorithms, to carry out research, as well as create Non-Player Characters, which perceive their surroundings, make decisions and take actions.

References

[1] C. Beattie, et al., DeepMind lab. arXiv:1612.03801, 2016.
[2] V. Mnih, et al., Human-level control through deep reinforcement learning. Nature 518, pp 529–533, 2015.
[3] M. Johnson, K. Hofmann, T. Hutton, & D. Bignell, D., The Malmo Platform for Artificial Intelligence Experimentation. In IJCAI pp. 4246-4247, 2016.
[4] M. Kempka, et al., Vizdoom: A doom-based ai research platform for visual reinforcement learning. In Computational Intelligence and Games (CIG), IEEE Conference, pp. 1-8, 2016.
[5] A. Juliani, V. Berges, E. Vckay, Y. Gao, H. Henry, M. Mattar, D. Lange., Unity: A General Platform for Intelligent Agents, arXiv:1809.02627v1, 2018.
[6] Venkataramanan. K, Game artificial intelligence literature survey on game AI, National University of Singapore, http://www.nus.edu.sg/nurop/2009/FoE/U058557W.PDF, 2008.
[7] AI Game Dev. (2008). Research Opportunities in Game AI. Retrieved January 2, 2008, from http://aigamedev.com/questions/research-opportunities, 2008.
[8] A. Narayek, "AI in computer games", Volume 1, Issue 10, Game Development, ACM, New York, February, 2004
[9] J. Wexler, "Artificial Intelligence in Games: A look at the smarts behind Lionhead Studio's 'Black and White' and where it can and will go in the future", University of Rochester, Rochestor, NY, 2002.
[10] Unity navigation system, https://docs.unity3d.com/Manual/nav-InnerWorkings.html.
[11] Unity NavMesh, NavMesh Agent, NavMesh Obstacle https://docs.unity3d.com/Manual/nav-NavigationSystem.html
[12] Get started with Mixed Reality https://docs.microsoft.com/en-us/windows/mixed-reality/