ICIC Express Letters
Volume **16**, Number **3**, March **2022**
 ISSN 1881-803X
pp. **299–306**

COMPARING PATHFINDING ALGORITHMS FOR INDOOR POSITIONING SYSTEM

Gede Putra Kusuma, Ricky Martin Goutama and Steven Ferdianto

Computer Science Department, BINUS Graduate Program – Master of Computer Science
Bina Nusantara University
Jl. K. H. Syahdan No. 9, Kemanggisan, Palmerah, Jakarta 11480, Indonesia
inegara@binus.edu; { ricky.goutama; steven.ferdianto }@binus.ac.id

Received April 2021; accepted July 2021

Abstract. *For many fields like robotics, automation, and video games, pathfinding is a fundamental issue. While outdoor pathfinding is carried out, due to the lack of indoor maps, indoor pathfinding remains a challenge. With various pathfinding algorithms and due to minimal resources of indoor maps, finding the most efficient algorithm for indoor pathfinding also is a challenge. This research compared several popular pathfinding algorithms like A* or A Star, REA* or Rectangle Expansion A* and Unity Pathfinding Package. The experiment also used Bina Nusantara Anggrek Campus 8th floor as the indoor map. The map was converted from a blueprint into 2D grid-style map. Total time and cost to find a path from a certain node to another node were calculated and repeated several times. When the total time and path cost are calculated for each algorithm, the results show that REA* has the shortest amount of time and path cost when determining a path.*
Keywords: Indoor pathfinding, Indoor navigation, A* algorithm, REA* algorithm, Unity pathfinding algorithm

1. **Introduction.** Pathfinding or pathing on 2D grid maps is a fundamental technology of video games and automation and robotics [1-3]. Pathfinding or pathing commonly refers to plotting the shortest route from starting point to ending point, while avoiding collision with the obstacles. With the advancement of algorithms and technology, it causes many new pathfinding algorithms to emerge. Some of the algorithms are still the same algorithm but have differences in preprocessing and optimizer methods. However, with many emerging pathfinding algorithms, each algorithm also has advantages and disadvantages for certain cases.

Most individuals have been enjoying the benefits from pathfinding algorithms especially outdoor pathfinding using map applications like Google Maps. However, indoor pathfinding remains challenging due to the lack of indoor maps, which are mainly plotted and modeled manually [4]. Indoor pathfinding is essential for indoor navigation applications, such as guiding visitors to reach point-of-interest inside a building or providing navigation for indoor butler robot.

In this research, we aim to find the most efficient algorithm for indoor pathfinding by comparing several popular pathfinding such as A* or A Star, REA or Rectangle Expansion A* and Unity Pathfinding Package. The comparison will be using Bina Nusantara Anggrek Campus 8th floor for the map. The map was also converted from blueprint into a 2D grid tile map. The pathfinding comparison experiment will be created and conducted using unity engine.

This paper is organized as follows. In Section 2, we introduce and describe related works in this field. Next, in the Proposed Method section, we describe and present the

DOI: 10.24507/icicel.16.03.299

proposed method. Furthermore, the experimental designs and results will be described and shown in the Experiments section. Finally, the last section presents the research conclusion and future research direction.

2. **Related Works.** There are a lot of previous works and various algorithms used in their works. In [5], the researchers offered 2 types of indoor routing, namely the feasible route and the comfortable route. This is because this paper aims to help people with disabilities use indoor routing as well. The feasible route is the proper and fastest route that can be used by everyone without disabilities, while the comfortable route can be used by everyone and people with disabilities. The difference between the routes is that for a comfortable route the generated path will use an elevator and for a feasible route, the generated path will tend to use stairs. The shortest path from the starting position to the destination is generated using the Dijkstra path finding algorithm where it will consider the elevator only (for comfortable route option) or elevator and stairs (for feasible route option) for every requested route. The result of this study is that the Dijkstra algorithm can generate the feasible and comfortable shortest route properly without any issues.

In [6], the algorithm used to create the navigation system is the A* algorithm. This research began by making a 2D map of the researcher's campus and then they implemented the A* algorithm and finally they tested it on an Android device. From the results of trials by researchers in this experiment, it was found that the accuracy of the A* algorithm tested on the researcher's campus layout was 87.75% of the 49 tests.

The researchers in [7] created a navigation system using their campus as their research object. They converted the blueprint of their campus building into a grid map and implemented an A* algorithm for the pathfinding method. The grid map they created consists of doors, corners of the room, stairs (treated as virtual doors), and floors that are all treated as a node. They created this navigation system on a web platform using JavaScript and they had tested the application for 75 times with setting up the starting point and goal point and then they will run the path finding algorithm and find its way. The result of this test is that they got a 100% accuracy and worked well.

[8] aims to create a robot that can carry humans and bring them to certain positions safely. The algorithm used for the navigation in this paper is A*, but there are slight modifications to the A* algorithm in the heuristic calculation section. To calculate the heuristic, they used equation

$$h(n) = k \times \min\left(\left|x_n - x_{goal}\right|, \left|y_n - y_{goal}\right|\right) \tag{1}$$

where k is the length of each grid in the grid environment, while x_n and y_n are the current robot positions and x_{goal} and y_{goal} are the destinations for the robot. Finally, the modified A* algorithm is compared to the A* algorithm. The results of this study show that the modified A* algorithm gained an increase in the time needed by 35%-47% and reduction in turning points by 50%-80%.

In [9], the researchers wanted to create an indoor navigation system using KLIA2 (Kuala Lumpur International Airport 2) as its layout and Dijkstra algorithm as its path finding method. The researchers used the second floor of KLIA2 that consists of 129 shop lots. First, they created a map using the floorplan with 1 : 350 scale and mapped all the shops into nodes that are connected to each other. Second, they tested the program 5 times to check if the prototype is working perfectly for the user by testing its functionality, and the results from these tests are working 100%. Third, they tested the reliability function by asking 3 respondents with 5 test cases to find a way from a determined starting point and ending point and the path they took will be recorded and compared to the generated path with the Dijkstra algorithm. The result from the third step is the path that the Dijkstra algorithm generated is more efficient than the path the respondents took.

[10] is an analysis paper that compared two shortest path algorithm that are Dijkstra algorithm and A* algorithm. The researchers used their town map and converted the map into a connected node. They tested the 2 path-finding algorithms by running those 2 algorithms on some test cases and the result is that both algorithms can find the shortest path but with slightly time difference, so it can be concluded that the performance is the same.

In [11] the research is about creating a navigation system for an indoor environment using Bluetooth beacon as its indoor positioning system. To determine the current position of the user, the researchers used the RSSI method that they got from the beacons placed inside the building. To navigate the user, the Dijkstra algorithm is chosen by the researcher as its path finding method. The result of this study is that the Dijkstra algorithm works well with the RSSI method with an error of 1m.

[12] created a wearable system to navigate visually impaired people. The positioning system is created by combining Radio Frequency Identifier (RFID) and Quasi-Zenith Satellite System (QZSS). The researchers also implemented HoloLens technology to generate a 3D mapping of the environment by actively detecting the surrounding objects. For the navigation system, the researcher combined the Dijkstra algorithm to calculate the global route and the A* algorithm for object avoiding navigation. The result is that the accuracy error of the positioning system is less than 1m, and the navigation system succeeds guiding a visually impaired person.

The proposed shortest path method in [13] is using a modified A* algorithm that is called rectangle expansion A* or REA* that is compared to the A* algorithm. The researchers created this modified A* algorithm to find a more efficient pathfinding algorithm in the grid map environment. They had tested and compared the REA* with A* algorithm with 12 different test cases in also 12 different grid maps. The result of this study is that REA* outperforms the A* algorithm in the time aspect when used in the grid map environment.

Based on the reviews of the above papers, the pathfinding algorithms are performing well and are easy to implement. Both Dijkstra and A* algorithm seems to be performed just slightly differently in a time aspect for a small test environment like building or town map. One of the papers also adopted the real blueprint of a building into a grid map and implemented the pathfinding algorithm, for example, in [6], they implemented their campus building blueprint into the grid map. The REA* algorithm is also interesting because it can outperform the A* algorithm in the grid map environment. Therefore, these various performance of pathfinding algorithms motivated authors to compare and find out the best algorithm to be implemented in the indoor positioning system with 2D grid map environment.

3. **Proposed Method.**

In this section, we described the pathfinding algorithms that we used in this experiment. We compared 3 pathfinding algorithms that are A* algorithm, REA* algorithm, and unity pathfinding algorithm to find the most efficient way to do a navigation for indoor positioning scenario.

3.1. **A Star algorithm.**

A* algorithm is one of the most famous pathfinding algorithms, and this algorithm will search for a path with the most minimum cost from the given starting node to its destination node [14]. To determine the order of which node will be visited, it uses a path and heuristic cost function (usually denoted as $f(n)$) [15]. The path and heuristic cost function consist of the summing of two functions that are the path cost function (usually denoted as $g(n)$) and heuristic cost function (usually denoted as $h(n)$). The past cost function is gained by calculating the cost to get to the n node from the starting node, while the heuristic cost function is gained by calculating the cost from the n node to its destination node. Therefore, the equation for this function is

$$f(n) = g(n) + h(n) \tag{2}$$

while the A* algorithm searching for the path through the graph, it follows the most minimum cost path and at the same time, it also saves the other possible node in a queue that is sorted by the cost ascendingly [16]. Also, while searching for the path, if it visited the node with the cost above the other that has been encountered, then it will abandon the cost that is above the other and start to search again through the node that has a lower cost. This process will keep repeating itself before it can find the destination node.

3.2. **Rectangle expansion A* algorithm.** Rectangle expansion A* or REA* is a modified A* algorithm that has the same goal as the normal A* that searches for the least cost path from the given starting node to its destination node. To use the REA* algorithm, we need to conduct it in the grid-based map because it is designed to work in the grid-based map. Each grid in the map consists of node n and node n consists of several variables like its coordinate that consists of x and y, *traversable* Boolean that determine if the node is walkable (the value will be true) or not (the value will be false), *gval* that is the cost from the starting node to the n node, *hval* that is the estimated cost from the n node to the destination node, *fval* that is the sum of *gval* and *hval* and mode to determine if a node has been visited or not. In the beginning, a global variable called *invmode* will be created and initialized its value as 0, and then the mode of each node in the map will also be initialized as 0. Any time a node receives a *hval*, the value of the node *mode* will be changed to *invmode*+2 and any time a node receives a *gval*, then the node *mode* will be changed into *invmode*+1. The *invmode* will also be modified every task is done by increasing its value by 3. If any node that has a *mode* less than *invmode*+1, then that node's *gval*, *hval*, and *fval* are invalid because it has not been visited yet [17]. To calculate the *gval* and *hval*, we used the octile distance function to calculate the distance between node n and node n', and the function is

$$octile\,(n, n') = 1.414 \times \min(\Delta x, \Delta y) + |\Delta x - \Delta y| \tag{3}$$

$$\Delta x = |x - x'| \text{ and } \Delta y = |y - y'| \tag{4}$$

First, the starting node (S) is going to scan both the horizontal and vertical directions until it stops because of an obstacle or the boundary of the map. After scanning the available direction, it will generate the new rectangle base on the scanned available node before. If the destination node (D) is in the generated rectangle, then the process will end and a path will be generated straight from the S node to the D node; in other way, by using octile distance from S node, all the boundaries of the generated rectangle *gval* will be updated. Then, by pushing itself out, the boundary of the created rectangle will attempt to create a new successor and the unblocked one will be the new successor. The new successor will also be given a *minfval* to determine the order of the inserted successors into the open list. The *minfval* can be acquired by the minimum of its parent *fval*. After that, the successor will be named Current Best Node (CBN) inside the open list and with the *minfval* that has the minimum value, and this CBN will vertically and horizontally search the map again and create a new rectangle. This process is repeated until the D node is located within the successor's current rectangle.

In this experiment, first we created the REA algorithm in a java language and after it is done and works well in java, we converted the REA algorithm into the C# language, so we can run it in unity with the other pathfinding algorithm. While creating the REA, in the expanding part or scanning both vertically and horizontally from the starting node or the successor node, we prioritize the scan in order from northwest, northeast, southwest, and finally southeast. So, if there is a possible way in the northwest side, then it will expand or search vertically and horizontally to the northwest side, otherwise; it will search by the order.

3.3. **Unity pathfinding algorithm.** Unity is a cross-platform game development engine created and maintained by Unity Technologies company. Unity engine is mainly used to create three-dimensional, two-dimensional, virtual reality, and augmented reality games, as well as simulations and other experiences. Unity engine also has been adopted by many industries outside video gaming, such as the movie industry, automotive, architecture, engineering, and construction.

Unity has an asset store where developers can download many packages of assets like sprites, animation, special effects, 3D models, and algorithms. One of the most popular packages is the pathfinding algorithm since almost every modern game has a pathing feature.

Unity pathfinding algorithm package is an innovative artificial intelligence pathfinding system without a navigation mesh. Feature a lot of different useful settings to let the pathfinder fit in any situation, programmable obstacles, static obstacles and useful debugging and testing features [18].

4. Experiments.

4.1. **Experimental design.** First, we create a grid map out of a floor plan blueprint of the Bina Nusantara University Anggrek Campus 8th floor as shown in Figure 1. We generated this grid map manually using unity by dividing the floor plan blueprint into 70×30 grids of width and height with the size of each grid around 80 cm $\times$ 80 cm in the real building environment. We also listed all the available routes and walls for each grid tile and manually assigned it into our grid map as shown in Figure 2.

Second, we created 50 test cases of starting node and target node for the pathfinding simulation and tested each of the test cases using A* algorithm, REA* algorithm, and

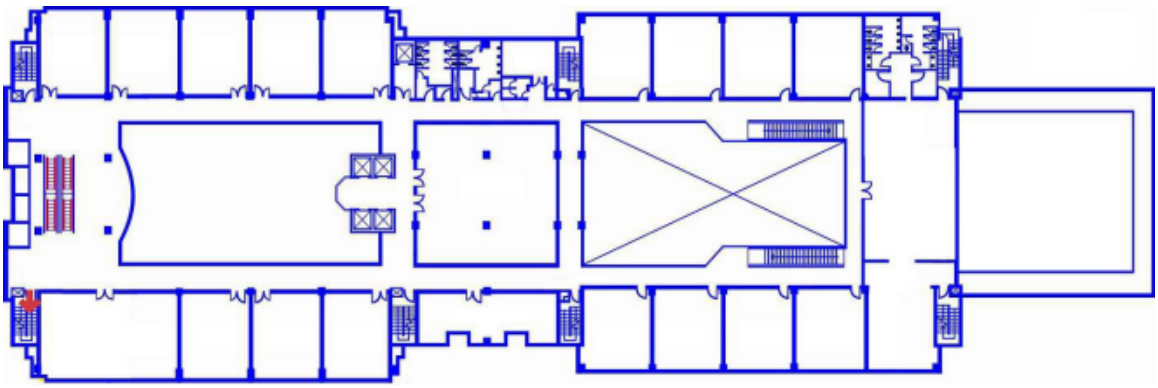

FIGURE 1. The blueprint of the Bina Nusantara University Anggrek Campus 8th floor

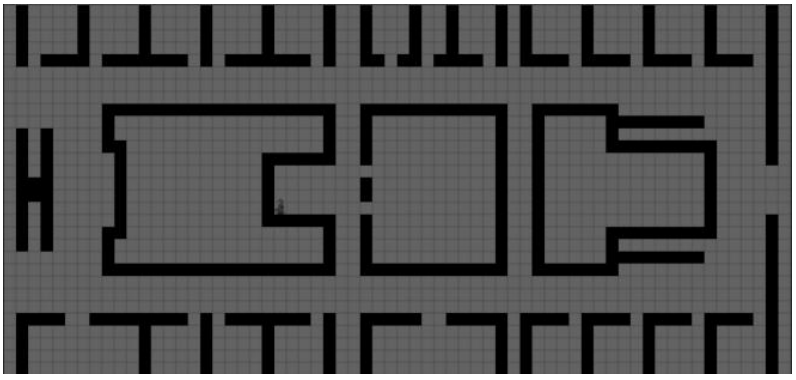

FIGURE 2. The generated map in the unity from the Bina Nusantara University Anggrek Campus 8th floor

unity pathfinding algorithm. We also used the same map and PC for each of the tests to make sure to provide consistent data. While we tested each of the algorithms, the time it took from the starting node to the target node was also registered and how much is the cost it took to find the path.

The example of one of the test cases that we have is shown in Figure 3, where the yellow circle is the starting node, and the red circle is the target node. The sequence of the test for each test case is first, to find the path from the starting node to the target node, we use the A* algorithm and record its time and total path cost, second, we use the REA* algorithm to find the path and record its data as the first one, and lastly, we use the unity pathfinding algorithm to find the path and record its data.

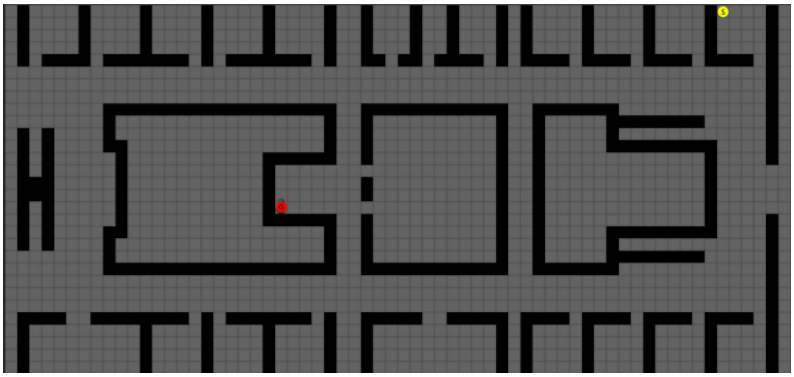

FIGURE 3. (color online) The example of a test case

4.2. **Experimental results.** Table 1 shows our results for the performance of the 3 path finding algorithms that we tested. All the results are acquired by recording the total path cost and the time of each test case. For the time, we calculate the delta time for each frame right before the algorithm starts running until the algorithm finishes or returns a list of nodes that contain a backtracked path. For the total path cost calculation, we calculate by summing all the path costs in the backtracked path. Since we are using a grid map to conclude this experiment, it is proven that REA* is the fastest because it searches for the target node on a large scale by expanding its search with rectangle area while the others search for the target node by opening one by one grid at the time.

TABLE 1. Summary of test results

Number of test cases	A*		REA*		Unity pathfinding	
	Path cost	Time (s)	Path cost	Time (s)	Path cost	Time (s)
1	38.40	0.0128	36.53	0.0093	38.80	0.0103
2	20.20	0.0943	19.81	0.0716	22.00	0.0858
3	38.40	0.0628	35.80	0.0134	39.20	0.0579
4	33.60	0.0546	30.43	0.0186	34.20	0.0410
5	59.40	0.0972	55.76	0.0692	60.20	0.0749
...	...	...	...	...	...	...
46	26.20	0.0177	25.31	0.0086	26.60	0.0120
47	38.00	0.0172	37.19	0.0113	39.00	0.0159
48	48.00	0.0159	45.78	0.0100	48.00	0.0118
49	30.80	0.0173	29.16	0.0070	31.40	0.0095
50	46.60	0.0135	42.97	0.0082	46.60	0.0097
Average	**37.30**	**0.0434**	**34.91**	**0.0296**	**37.50**	**0.0355**

5. **Conclusion and Future Work.** This research aims to show the differences in the performance and compare various pathfinding algorithms, so the result can become a reference and benchmark in making applications, especially in indoor positioning system. Based on the data from Table 1, we can conclude that the REA* is the fastest algorithm in finding the target point among the others. Other than that, it also has the smallest average of path cost among the others too. The second best one is the unity pathfinding algorithm which has the second smallest average time, but the average path cost is variative compared to the other two algorithms. The last one is A* which has the worst average time but also is second best in terms of average path cost. Therefore, REA* is the best out of these 3 algorithms and works well in the converted Bina Nusantara University Anggrek Campus 8th floor.

For future research, different pathfinding algorithms should be tested and compared. The algorithms also need to be tested in different map models and different indoor environments. Another point to be researched is that the map should be expanded into a multi-level map, so it can move from another level into another level inside the buildings. The different approach of saving the calculated data and pre-calculating certain values to minimize the pathfinding complexity could also be interesting for a future topic.

REFERENCES

[1] Z. A. Algfoor, M. S. Sunar and H. Kolivand, A comprehensive study on pathfinding techniques for robotics and video games, *International Journal of Computer Games Technology*, pp.1-11, DOI: 10.1155/2015/736138, 2015.

[2] Z. Yao, W. Zhang, Y. Shi, M. Li, Z. Liang and Q. Huang, ReinforcedRimJump: Tangent-based shortest-path planning for two-dimensional maps, *IEEE Trans. Industrial Informatics*, vol.16, no.2, pp.949-958, DOI: 10.1109/tii.2019.2918589, 2020.

[3] H. Cheng, H. Chen and Y. Liu, Topological indoor localization and navigation for autonomous mobile robots, *IEEE Trans. Automation Science and Engineering*, vol.12, no.2, pp.729-738, DOI: 10.1109/tase.2014.23, 2015.

[4] X. Zhou, Q. Xie, M. Guo, J. Zhao and J. Wang, Accurate and efficient indoor pathfinding based on building information modelling data, *IEEE Trans. Industrial Informatics*, vol.16, no.12, pp.7459-7468, DOI: 10.1109/TII.2020.2974252, 2020.

[5] P. M. Dudas, M. Ghafourian and H. A. Karimi, ONALIN: Ontology and algorithm for indoor routing, *2009 10th International Conference on Mobile Data Management: Systems, Services and Middleware*, pp.720-725, 2009.

[6] S. Kasim, L. Y. Xia, N. Wahid, M. Md Fudzee, H. Mahdin, T. Ramli, S. Suparjoh and M. Salamat, Indoor navigation using A* algorithm, in *Recent Advances on Soft Computing and Data Mining. SCDM 2016. Advances in Intelligent Systems and Computing*, T. Herawan, R. Ghazali, N. M. Nawi and M. M. Deris (eds.), DOI: 10.1007/978-3-319-51281-5_60, 2017.

[7] A. Maulana and W. Wijanarto, Implementation of A* algorithm in a web-based application to find the shortest route as indoor digital map navigation (Implementasi algoritma A* dalam aplikasi berbasis web untuk menemukan rute terpendek sebagai navigasi peta digital indoor), *Creative Information Technology Journal*, vol.5, no.1, DOI: 10.24076/citec.2017v5i1.129, 2019.

[8] H. Yu, X. Miao, S. Wang and Y. Hu, Nursing robot safety path planning based on improved A Star algorithm, *Journal of Compurters*, vol.30, no.3, DOI: 10.3966/199115992019063003023, 2019.

[9] K. A. F. A. Samah, A. Sharip, I. Musirin, N. Sabri and M. Salleh, Reliability study on the adaptation of Dijkstra's algorithm for gateway KLIA2 indoor navigation, *Bulletin of Electrical Engineering and Informatics*, vol.9, no.2, DOI: 10.11591/eei.v9i2.2081, 2020.

[10] D. Rachmawati and L. Gustin, Analysis of Dijkstra's algorithm and A* algorithm in shortest path problem, *Journal of Physics: Conference Series*, DOI: 10.1088/1742-6596/1566/1/012061, 2020.

[11] R. Jyothis, Aathira, S. Ashiq and G. Ramesh, Indoor navigation based on Bluetooth beacons, *Proceedings of the International Conference on Microelectronics, Signals and Systems*, DOI: 10.1063/5.0004408, 2019.

[12] A. Yamashita, K. Sato and K. Matsubayashi, Walking navigation system for visually impaired people based on high-accuracy positioning using QZSS and RFID and obstacle avoidance using HoloLens, *International Journal of Innovative Computing, Information and Control*, vol.16, no.4, pp.1459-1467, 2020.

[13] A. Zhang, C. Li and W. Bi, Rectangle expansion A* pathfinding for grid maps, *Chinese Journal of Aeronautics*, vol.29, no.5, pp.1385-1396, DOI: 10.1016/j.cja.2016.04.023, 2016.
[14] K. Khantanapoka and K. Chinnasarn, Pathfinding of 2D & 3D game real-time strategy with depth direction A* algorithm for multi-layer, *2009 8th International Symposium on Natural Language Processing*, pp.184-188, 2009.
[15] P. E. Hart, N. J. Nilsson and B. Raphael, A formal basis for the heuristic determination of minimum cost paths, *IEEE Trans. Systems Science and Cybernetics*, vol.4, no.2, pp.100-107, DOI: 10.1109/TSSC.1968.300136, 1968.
[16] W. Y. Loong, L. Z. Long and L. C. Hun, A Star path following mobile robot, *2011 4th International Conference on Mechatronics (ICOM)*, pp.1-7, 2011.
[17] L. Chong, J. Li and X. Liu, Static rectangle expansion A* algorithm for pathfinding, *IEEE Trans. Games*, DOI: 10.1109/TG.2020.3012602, 2020.
[18] Unity Technologies, *Language*, https://assetstore.unity.com/account/assets, Accessed on February 22, 2021.