IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2320-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 10, Issue 5, May 2021

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.512

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

||Volume 10, Issue 5, May 2021||

DOI:10.15680/IJIRSET.2021.1005099

Game Development Using Behaviour Tree in Unreal Engine 4

Ritik Kothari[1], Smit Nawar[2], Siddharth Kothari[3], Jaya Jeswani[4]

Under Graduate Student, Department of Information Technology, Xavier Institute of Engineering, Mumbai, India[1-3]

Assistant Professor, Department of Information Technology, Xavier Institute of Engineering, Mumbai, India.[4]

ABSTRACT: Artificial intelligence has been a growing resource for video games for years now. Most video games, whether they're racing games, shooting games, or strategy games, have various elements that are controlled by AI, such as the enemy bots or neutral characters. Even the ambiguous characters that don't seem to be doing much are programmed to add more depth to the game, and to give you clues about what your next steps should be. AI in video games is a distinct subfield and differs from academic AI. It serves to improve the game-player experience rather than machine learning or decision making. During the golden age of arcade video games, the idea of AI opponents was largely popularized, in the form of graduated difficulty levels, distinct movement patterns, and in-game events dependent on the player's input. AI is often used in mechanisms which are not immediately visible to the user, such as data mining and procedural-content generation. However, there are also many other ways that AI and game development are growing through each other-er. Although AI continues to be used to bring video games to life, video games are now being trained to study their own patterns so as to improve their own algorithms, which is just one of many ways that AI is becoming more advanced.

KEYWORDS: Artificial Intelligence, AI Agents, Games, Behaviour Tree, Blackboard, Non-Player Character**.**

I. INTRODUCTION

The term "Game AI" is used to refer to a broad set of algorithms that include techniques from control theory, robotics, computer graphics and computer science in general, and so video game AI may often not constitute "True AI" in that such techniques do not necessarily facilitate computer learning or other standard criteria, only constituting "automated computation" or a predetermined and limited set of responses to a predetermined and limited set of inputs.[1][2][3] Game AI/heuristic algorithms are used in a wide variety of quite disparate fields inside a game.

The most obvious is in the control of any NPCs in the game, although "scripting" (decision tree) is currently the most common means of control.[4] These handwritten decision trees often result in "artificial stupidity" such as repetitive behaviour, loss of immersion, or abnormal behaviour in situations the developers did not plan for.[5] Pathfinding, a common use for AI, is widely seen in real-time strategy games. Pathfinding is the method for determining how to get a NPC from one point on a map to another, taking into consideration the terrain, obstacles and possibly "fog of war".[6][7] Commercial videogames often use fast and simple "grid-based pathfinding", wherein the terrain is mapped onto a rigid grid of uniform squares and a pathfinding algorithm such as A* or IDA* (graph traversals) is applied to the grid.[8][9][10] Instead of just a rigid grid, some games use irregular polygons and assemble a navigation mesh out of the areas of the map that NPCs can walk to [8][11].

As a third method, it is sometimes convenient for developers to manually select "waypoints" that NPCs should use to navigate; the cost is that such waypoints can create unnatural-looking movement. In addition, waypoints tend to perform worse than navigation meshes in complex environments.[12][13] Beyond static pathfinding, navigation is a subfield of Game AI focusing on giving NPCs the capability to navigate in a dynamic environment, finding a path to a target while avoiding collisions with other entities. Rather than improve the Game AI to properly solve a difficult problem in the virtual environment, it is often more cost-effective to just modify the scenario to be more tractable. If pathfinding gets bogged down over a specific obstacle, a developer may just end up moving or deleting the obstacle.[14].

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

||Volume 10, Issue 5, May 2021||

DOI:10.15680/IJIRSET.2021.1005099

II. RELATED WORK

Game developers have been working to develop more dynamic and credible NPCs with richer behaviour. The approach they found was to change the gameplay, so that the NPCs could also be improved.

2.1 Behaviour Trees

In Unreal Engine 4, Behaviour Trees are used to generate artificial intelligence (AI) in projects that include non-player characters. Although the Behaviour Tree asset is used and execute logic-containing branches to decide which branches should be executed, it depends on an asset called a Blackboard which serves as the 'brain' for a behaviour tree. The behaviour tree is passed downwards from the highest mode. After execution, the different Node types will return one of three states: successful, failure or running. The behavioural trees in commercial games have been widespread.

2.2 Blackboards

The Unreal Engine 4 blackboard asset includes a number of user-defined keys containing knowledge used by behaviour tree for decision making. For instance, CanSeePlayer is a Boolean key that can be referenced by the Behaviour Tree for the change of the value. If the value is true, the AI guard can execute a branch that causes it to chase the player character. If it is false, if the AI guard patrols across the environment triggered by another branch.

2.3 Environment Query System

The Environmental Query System (EQS) is a feature of the Unreal Engine 4 artificial intelligence system that collects environmental data. A Behaviour Tree can be used to called an EQS query and used to decide how the test outcomes can be de-pendent on it. EQS queries consist mainly of generators and Contexts. EQS queries are used to guide AI characters to locate the safest place for a player in order to strike, to catch health or ammunition, or to find the nearest point of cover.

III. IMPLEMENTATION

3.1 Player Chasing Branch on Behaviour Tree

The AI guard is controlled by a module called as AIController. For the guard to see and hear to the player character, a very important component is to be added which is called as AIPerception. After adding this com-ponent, we need to add two functions called as AI Sight Config and AI Hearing Config which can be tweaked accordingly. This function will help the AI Guard to see and hear the player character.

In the blueprint of the player character, we will need to add a component called AIPerception Stimulus Source and tweak few settings like check the Auto register as the source and add two senses called as AISence_Hearing and AISense_Sight.

The first step towards implementing behaviour tree is create two assets called Behaviour Tree and Blackboard. Then attach the behaviour tree to the AIController blueprint as shown below.

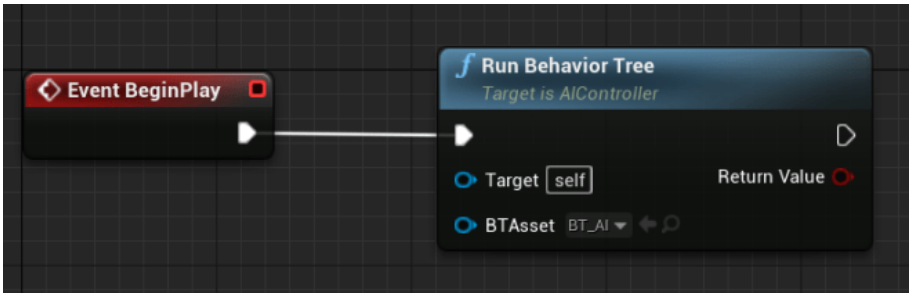


Figure 1: AIController Blueprint

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

||Volume 10, Issue 5, May 2021||

DOI:10.15680/IJIRSET.2021.1005099

Starting with basic implementation of AI guard, we implemented a function where, if the player character is in the line of sight of the AI guard, then it will chase the player character until it loses the sight of the player character. For that we need to create a task which help the AI guard to find the player location and another task which will make to move on the location received.

For finding the player character we will need to find its current location in the environment which will be provided by an in-built UE4 function called GetActorLocation. This function will fetch the location on the environment via another function called GetRandomPointInNavigableRadius. Then it will pass the location to a blackboard key called as TargetLocation. Then another task is created which will tell the AI Guard to move to the TargetLocation.

At last, we will put two decorators on the main node. A decorator is basically a condition which will allow the traversal in to the node only when that condition is met. For this branch the traversal will enter this branch only when the AI Guard can see the player character. In the condition the value of the Boolean blackboard key CanSeePlayer will be checked. If it is set then only it will enter the node and if it is not set it will move to another node.

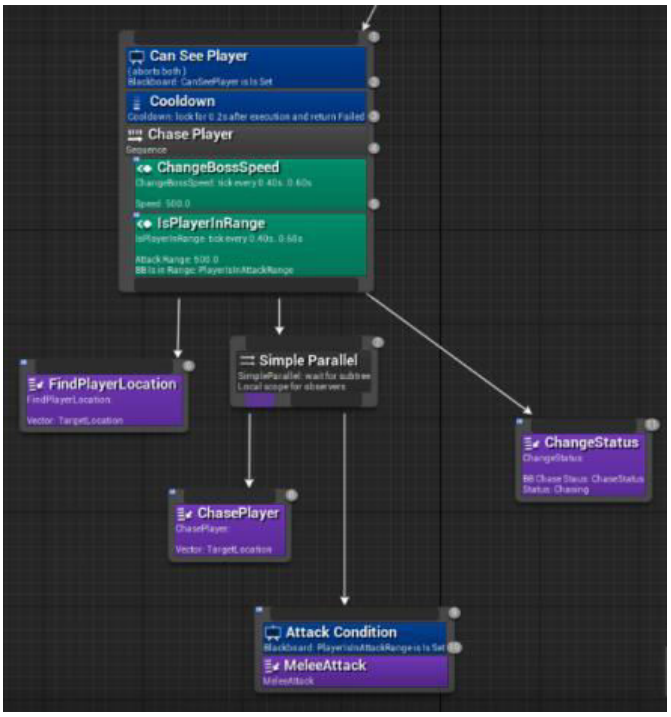


Figure 2: Behaviour Tree branch for Chasing the player.

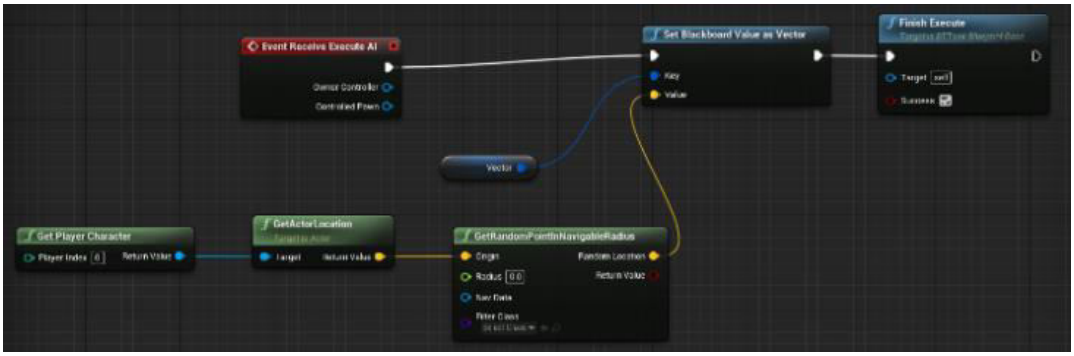


Figure 3: Blueprint to find player location.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

||Volume 10, Issue 5, May 2021||

DOI:10.15680/IJIRSET.2021.1005099

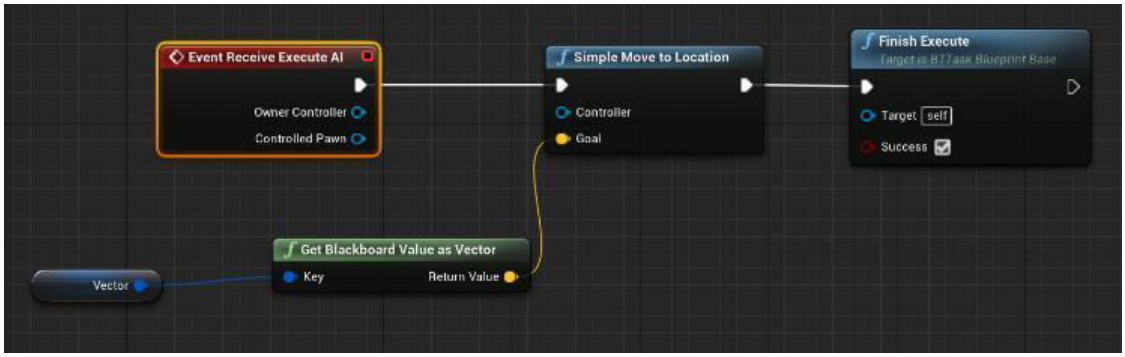


Figure 4: Blueprint to move to the given location.

3.2 Patrolling Branch on Behaviour Tree

After losing sight of the player character following the execution of the chasing branch, the AI guard will patrol on a fixed route. For that we will need to get the locations of the patrol points. Fir this a new task is created called as FindPatrolPoints. Here the location of patrol points will be given to a blackboard key called as PatrolPathVector.

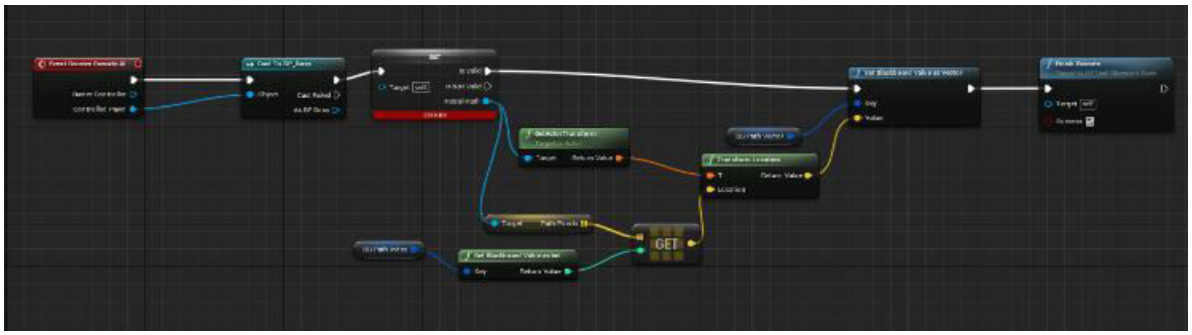


Figure 5: Blueprint for finding the patrol points.

After getting the points, now the behaviour tree will instruct the AI Guard to move to the location of patrol points stored in the blackboard key PatrolPathVector through a node called Move To and then it will wait there for the 2 seconds. In order to move to the next patrol point, we need to create another task called IncrementPathIndex, which will update the next location to move to. In this task the index of the patrol point will be incremented by 1 and the PatrolPathVector will be updated.

Now, For AI Guard to continue travelling the patrol points we need to put a loop so that it will patrol until it sees the player character. For that a task is created called LoopPath. Here it will check the value of a Boolean blackboard key called IsLooping. If the value is true, it will keep repeating the patrol route and if its false it will stop doing that.

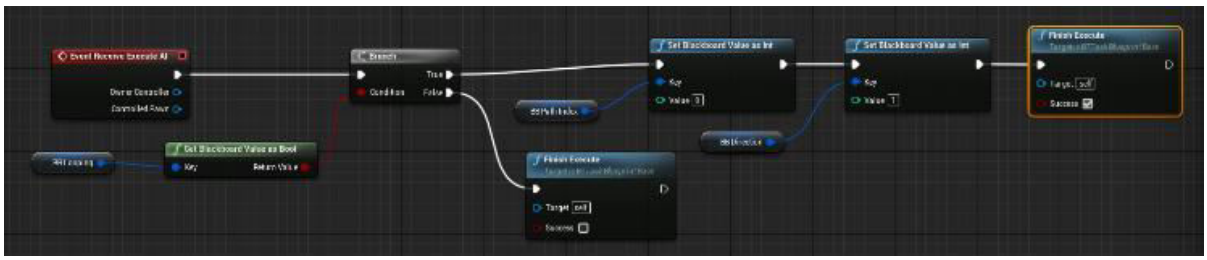


Figure 6: Blueprint for path looping.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

||Volume 10, Issue 5, May 2021||

DOI:10.15680/IJIRSET.2021.1005099

Now, once the guard reached last the patrol point it should return back to the starting point following the same path. For this duplicate the IncrementPathIndex blueprint and instead of incrementing by 1 we will decrement by 1.

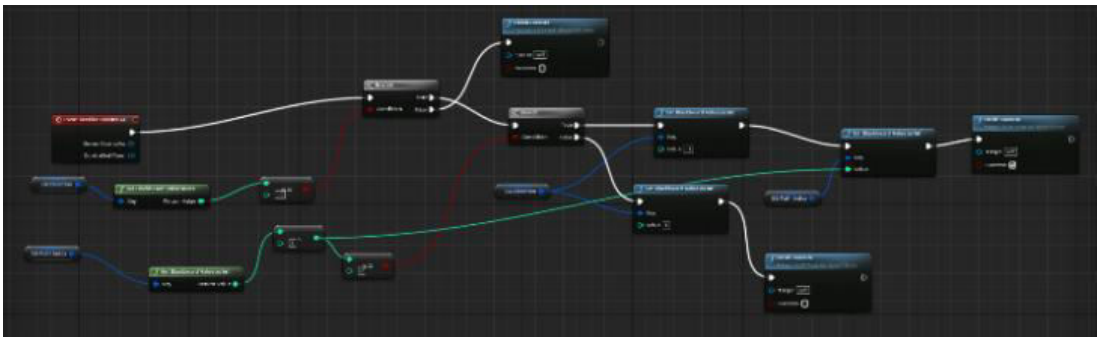


Figure 7: Blueprint for decrementing path index.

In order to limit the movement speed of the AI Guard while patrolling, we will add a create service called ChangeBossSpeed, which will reduce while patrolling and will increase while chasing the player character. For this we will create a new blackboard key of class vector and name it speed. In the blueprint we will simply limit the max walk speed to the value of the variable speed.

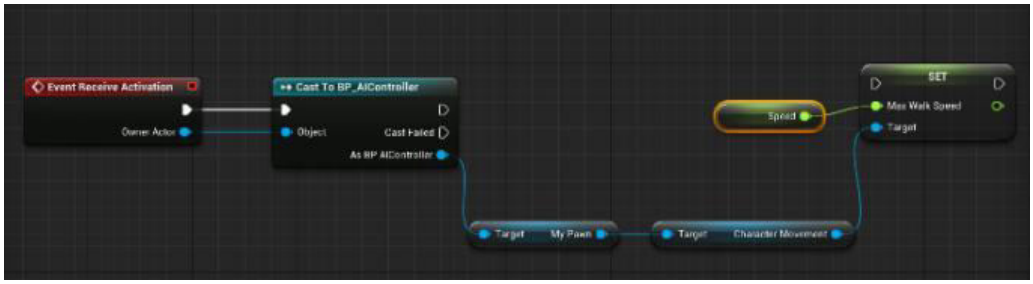


Figure 8: Blueprint for changing the max walk speed.

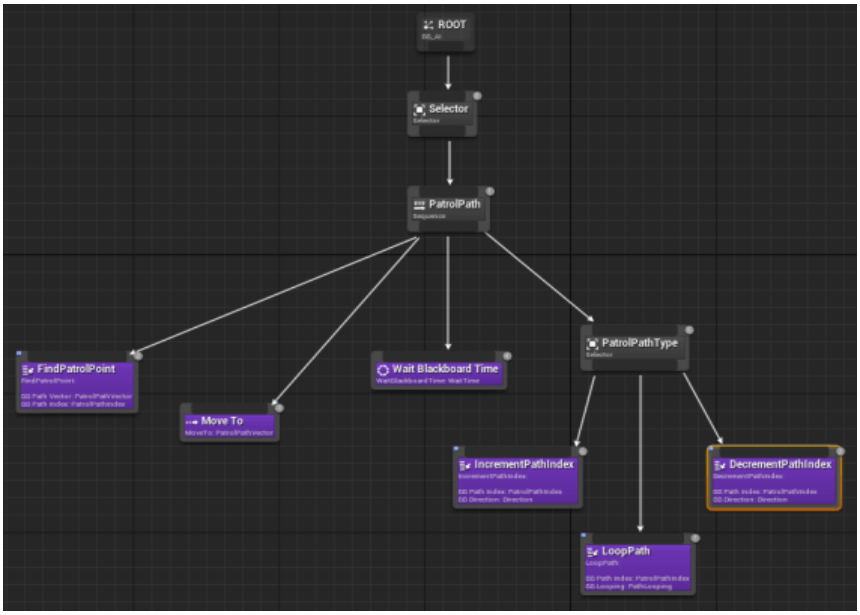


Figure 9: Behaviour Tree Branch for Patrolling.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

||Volume 10, Issue 5, May 2021||

DOI:10.15680/IJIRSET.2021.1005099

3.3 Searching Branch on Behaviour Tree

While chasing the player character, if the AI Guard loses the sight of the player character it will search for the player character in the close vicinity. It will get the last known position of the player form a task called SearchLocation. In this Blueprint the function GetActorLocation will search for the location in the environment then update the location in vector variable called TargetLocation. Then it will move to the location store in TargetLocation and wait for two seconds.

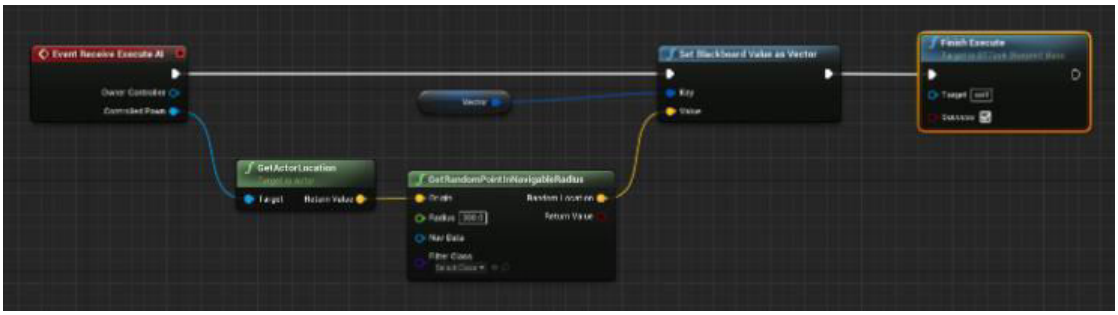


Figure 10: Blueprint for getting search location.

The searching node is a child node of the sequence node. It will enter the only when AI Guard cannot see the player character. For that we will add a decorator on the searching node and it will check the value of CanSeePlayer. Also, we will add a decorator called Loop which will allow the same node to repeat 3 times.

We will add a decorator on the main sequence node will all the traversal to the enter the only when the ChaseStatus in set to searching. ChaseStatus is an enumeration asset with three different states: Patrolling, Searching and Chasing.The ChaseStatus is updated in each brand through a task called ChangeStatus. Here the status is updated and stored in the variable ChaseStatus.

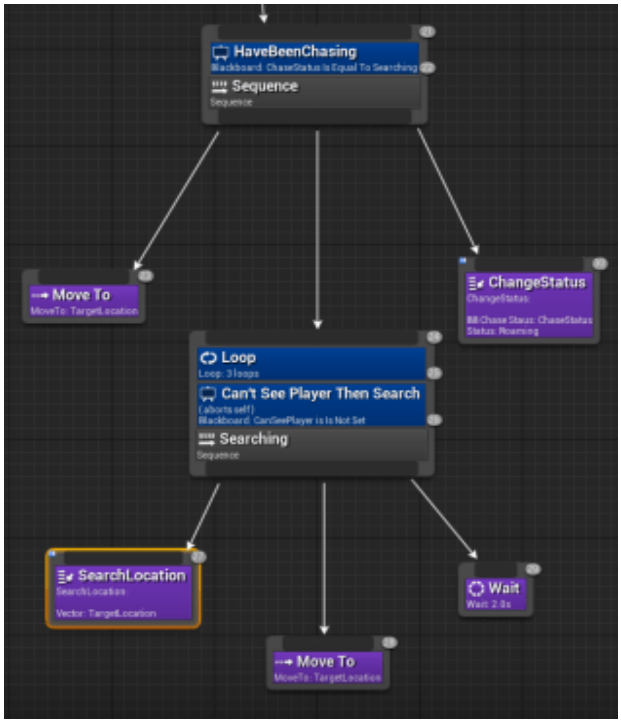


Figure 11: Behaviour Tree Branch for Searching.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

||Volume 10, Issue 5, May 2021||

DOI:10.15680/IJIRSET.2021.1005099

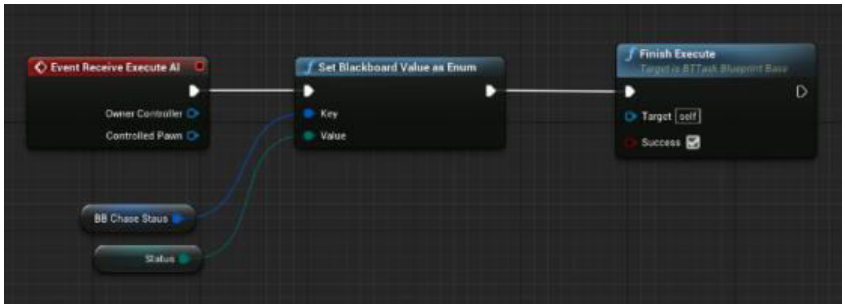


Figure 12: Blueprint for getting the status.

3.4 Investigation Branch on Behaviour Tree

In this branch, the AI Guard will move to a location from where the sound came and investigate the location. The traversal will only enter this node when the value of Boolean key IsInvestigating is set.In this branch the AI guard will move to the TargetLocation from where the sound came wait there for 2 seconds and look around. And if it doesn't find anything it will stop investigating and move back to patrolling.

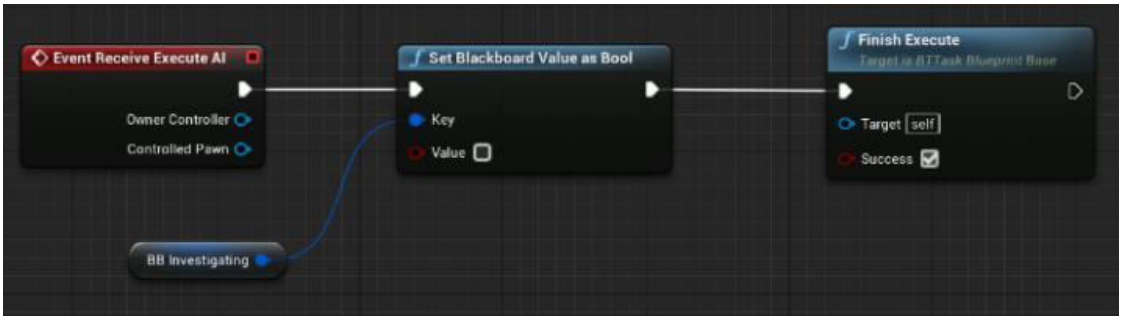


Figure 13: Blueprint to stop investigation.

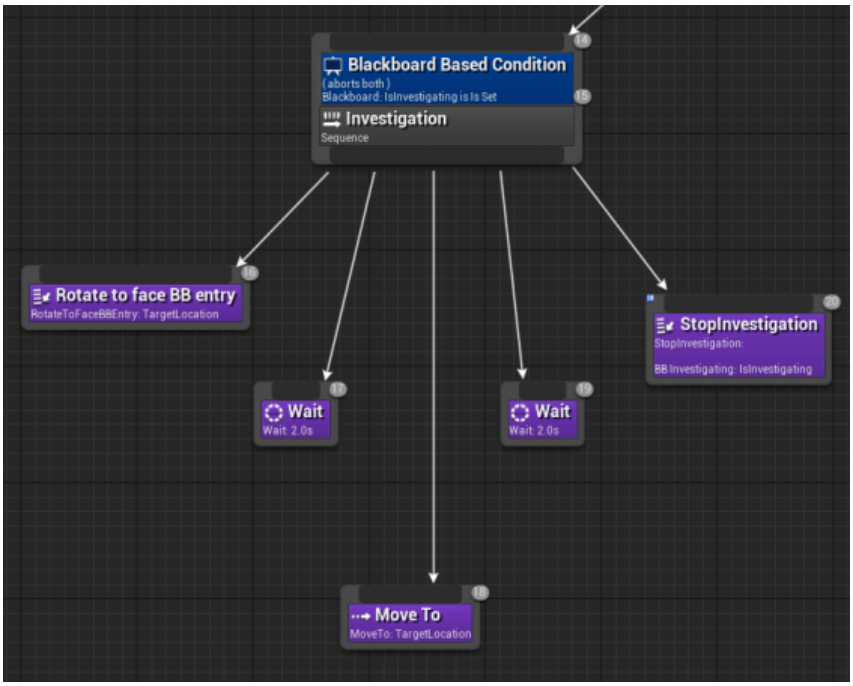


Figure 14: Behaviour Tree Branch for Investigation.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

||Volume 10, Issue 5, May 2021||

DOI:10.15680/IJIRSET.2021.1005099

3.5 Extending the Player Chasing Branch on Behaviour Tree

While chasing the player character, the AI Guard can perform an attack sequence when in certain range. This range will be determined by a service called IsPlayerInRange. In this blue print the distance between the controlled pawn i.e., the AI Guard and the player character is calculated and is compared to the value store in a variable called AttackRange. And update the value of Boolean variable called IsInRange. If the value is true, the attack node will be executed and if it is false then it will not be executed.

To perform the attack, we need to create a new task called MeleeAttack. In this task we will implement an attack interface which will provide the type of attack is going to perform and put it as a condition. If the condition is true then it will execute the node with a delay of 1.2 seconds.

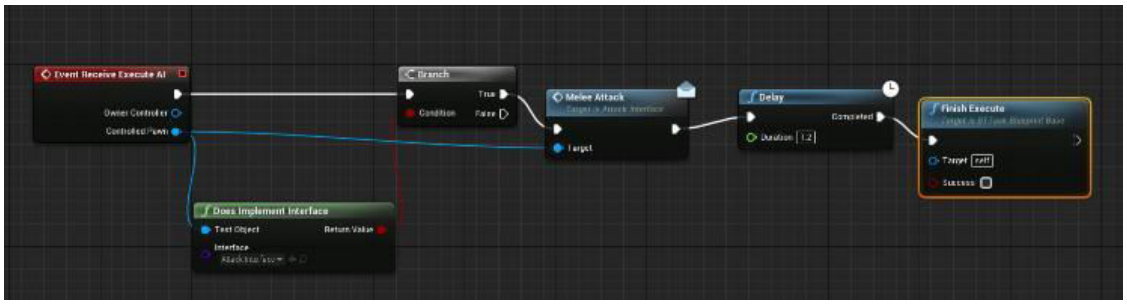


Figure 15: Blueprint for Melee Attack.

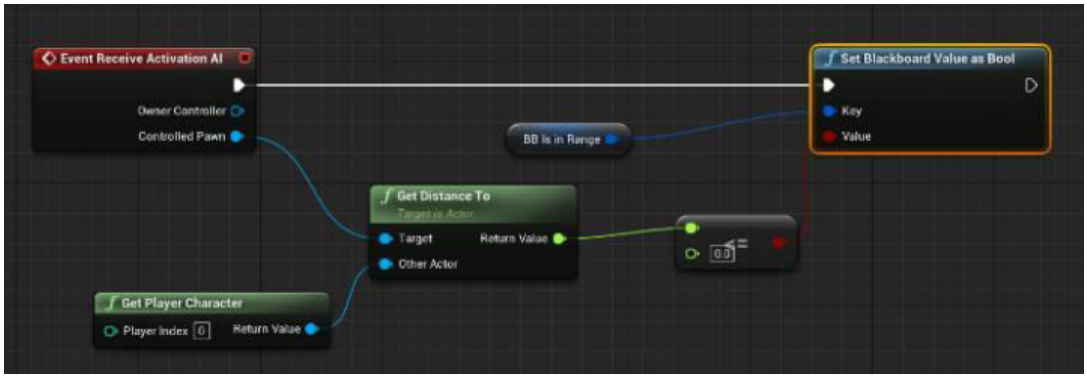


Figure 16: Blueprint to check if AI Guard is in range.

For AI Guard to chase and attack simultaneously, we have to this both tasks i.e., ChasePlayer and MeleeAttack as the child node to Simple parallel node. This node allows two tasks to perform simultaneously. The purple area represents the primary task and the black area represents the secondary task.

On the main Chase Player node, we will add the service called IsPlayerInRange which will verify wo variables i.e., AttackRange and a Boolean key PlayerIsInAttackRange. Similarly, we will add a decorator on the MeleeAttack node to check if the player is in range to attack

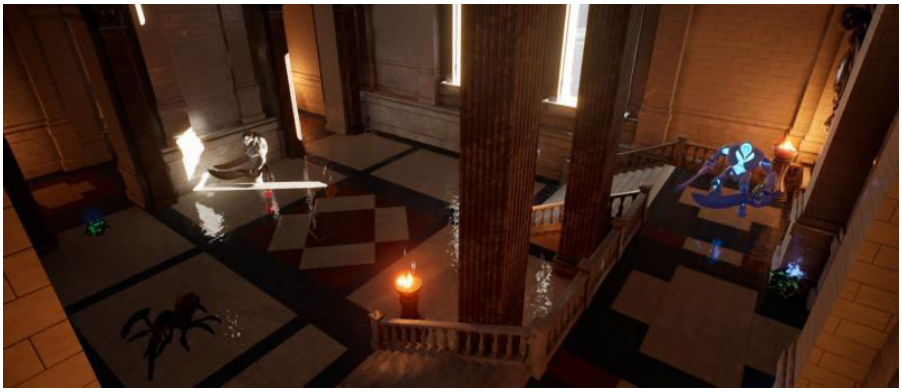
| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

||Volume 10, Issue 5, May 2021||

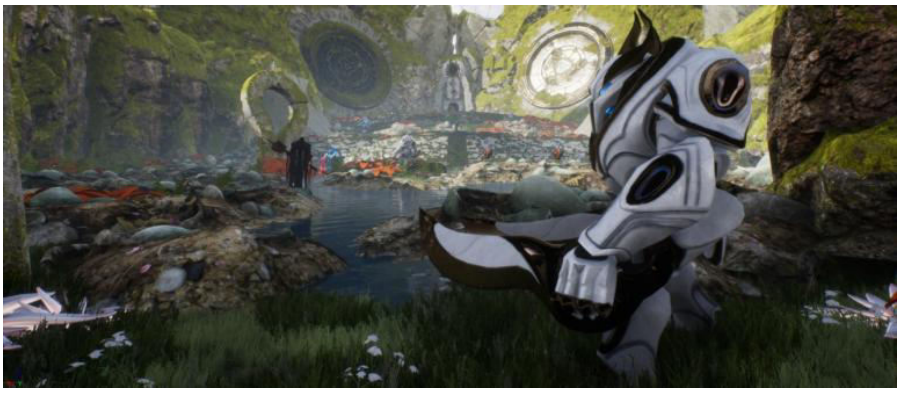
DOI:10.15680/IJIRSET.2021.1005099

IV. RESULTS

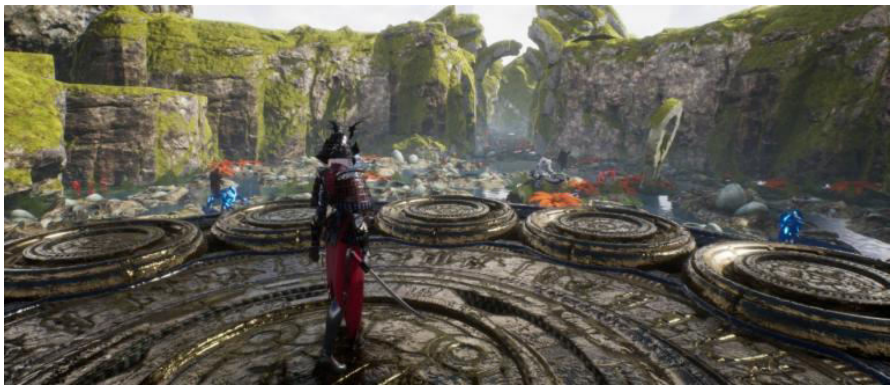
Figures shows the results of the game that is developed using behaviour trees in Unreal Engine 4.



(a)



(b)



(c)

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

||Volume 10, Issue 5, May 2021||

DOI:10.15680/IJIRSET.2021.1005099

(d)

Fig. 17: Results (a) Enemy characters – White Minion, Blue Minion and Spider (b) Enemy characters - De-Saad and Golden Minion. (c) Enemy boss - Countess (d) Enemy boss - Grux.

V. CONCLUSION

Artificial Intelligence adds realism to the gaming experience and make the game more powerful and meaningful. This RPG game is a prototype of using AI in Unreal Engine 4. Here, this game simulates combat from a third-person perspective and player will encounter op-ponent waves they need to survive in order to progress. AI can bring the revolution in the gaming industry. This project shows the basic use of AI in a game. Unreal Engine consist of many different functions which can be used in implementing advanced AI.

REFERENCES

[1] Bogost, Ian (March 2017). ""Artificial Intelligence" Has Become Meaningless". Retrieved 19 May 2020.
[2] Kaplan, Jerry (March 2017). "AI's PR Problem". MIT Technology Review.
[3] Eaton, Eric; Dietterich, Tom; Gini, Maria (December 2015). "Who Speaks for AI?" (PDF). AI Matters.
[4] Good, Owen S. (5 August 2017). "Skyrim mod makes NPC interactions less scripted, more Sims-like". Polygon. Retrieved 19 May 2020.
[5] Lara-Cabrera, R., Nogueira-Collazo, M., Cotta, C., & Fernández-Leiva, A. J. (2015). Game artificial intelligence: challenges for the scientific community.
[6] Yannakakis, G. N. (2012, May). Game AI revisited. In Proceedings of the 9th conference on Computing Frontiers (pp. 285–292). ACM.
[7] Hagelback, Johan, and Stefan J. Johansson. "Dealing with fog of war in a real-time strategy game environment." In Computational Intelligence and Games, 2008. CIG'08. IEEE Symposium On, pp. 55-62. IEEE, 2008.
[8] Abd Algfoor, Zeyad; Sunar, Mohd Shahrizal; Kolivand, Hoshang (2015). "A Comprehensive Study on Pathfinding Techniques for Robotics and Video Games". International Journal of Computer Games Technology. 2015: 1–11. doi:10.1155/2015/736138.
[9] Yap, Peter. "Grid-based path-finding." In Conference of the Canadian Society for Computational Studies of Intelligence, pp. 44-55. Springer, Berlin, Heidelberg, 2002.
[10] Sturtevant, N. R. (June 2012). "Benchmarks for Grid-Based Pathfinding". IEEE Transactions on Computational Intelligence and AI in Games. 4 (2): 144–148. doi:10.1109/TCIAIG.2012.2197681.
[11] Goodwin, S. D., Menon, S., & Price, R. G. (2006). Pathfinding in open terrain. In Proceedings of International Academic Conference on the Future of Game Design and Technology.
[12] Nareyek, A. (2004). AI in computer games. Queue, 1(10).
[13] https://docs.unrealengine.com/enUS/InteractiveExperiences/ArtificialIntelligence/BehaviourTrees/index.html (Unreal Engine 4 Documentation)
[14] https://docs.unrealengine.com/en-US/InteractiveExperiences/ArtificialIntelligence/EQS/index.html (Environment Query System – Unreal Engine 4 Documentation)

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor:
7.512

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर
NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details