3D Game based on FPS using Artificial Intelligence

Sanjay Kumar S[1], Sagar T Y[2], Jayanth C[3], Amruta T Bhat[4], Prof. Vinayashree[5]
[12345]*Vidya Vikas Institute of Engineering and Technology Mysuru,India*

Abstract- A game is a structured form of play. A new solution for the building a first- person shooter game is by using cross platform unity's game engine. This project uses unity's mecanium system in conjunction with root motion animation and navigation mesh based pathfinding mechanism, and some advanced methods like building the environment using navigation and pathfinding. It also uses animation state machines to control the animations of players and non-player characters. Player finds the path to the target in game scene. Another solution is A* algorithm is most widely used to estimate the distance of any point to the target point. 3-Dimensional rendering of the game objects is the interesting and attracting key of this project.

Keywords- unity's mecanim, pathfinding, a*algorithm, state machines, rendering.

I. INTRODUCTION

An *understanding* of graphs and algorithm is an integral part of the process. Establishing a clear understanding of object and how various algorithms affect those objects is crucial for anyone interested in learning programming. An algorithm is a procedure of operations for correctly solving a problem in a finite number of steps, where operations in each step can be executed in sequential order or in parallel. A graph algorithm, in turn, is an algorithm for solving problems that can be formulated in terms of graph and its related data structures.

II. A* ALGORITHM

A* algorithm is a shortest path heuristic search algorithm, and it is based on "Dijkstra" algorithm which finds the shortest paths from the source vertex to all other vertices in the graph. A heuristic is used to determine which neighbour to search next. This significantly speeds up the search by trying to visit as few nodes as possible. It does this by scoring neighbours as they are encountered based on their likeliness to lay on the cheapest path. It will always find the cheapest path .

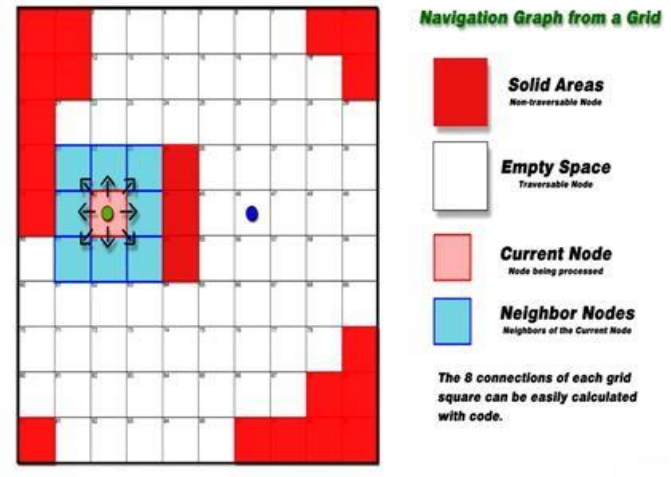


Fig.1: Navigation graph from a grid

A* search is a loop which maintains two lists namely open list and closed list. Open list: It contains nodes we have discovered as neighbours during the search but have not yet processed. It selects one node to process from the open list with each iteration of the search loop and remove it from the open list once processed. In the first loop iteration, the open list contains only the start node.

Closed list: As A* selects nodes from the open list, it removes them from the open list and it adds to the closed list. This allows to keep track of all the nodes that A* have already processed so it won't try to process them again (to avoid infinite loop). The closed list will start off completely empty.

Open list < 31, 51, 62, 63, 22, 23, 24, 50, 61, 21, 63, 65, 73, 74, 45, 56, 66 >

Closed list < 42, 43, 53, 33, 41, 52, 32, 64, 55, 46 >

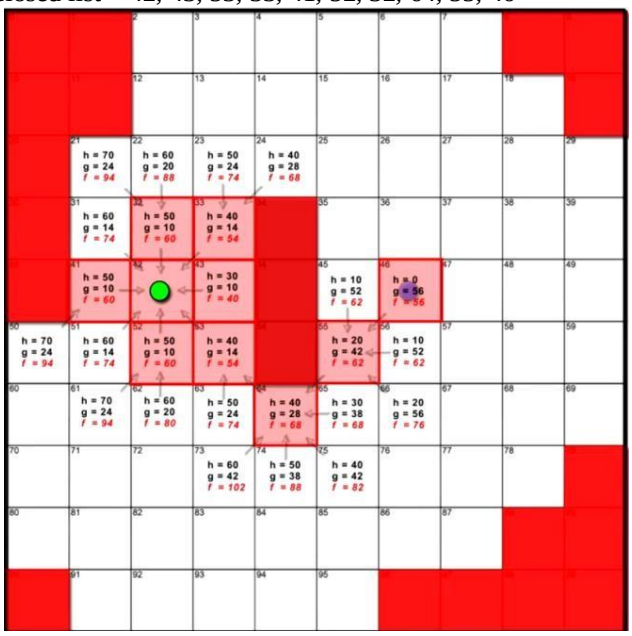


Fig.2: A* path finding

For Navigation and Path-finding problem, we have used the Manhattan Method of estimation to estimate the cost. The function of current node n is expressed as follows:

F(n)=H(n)+G(n)

Where, F(n) is a cost estimated function, H(n) is the heuristic value of the shortest path of any node n to the target point, G(n) is the path which is shortest and also the initial point to any node n.

Processing of a Neighbour in A* search

- In the iteration of search there is a calculation of score for the A* terminology and it is called as fScore of the node.
- A* will set its parent node reference to the current node that is going to be processed.
- If Neighbour already exists in Closed list the iteration loop will skip that Neighbour and does not add to Open list.

- If Neighbour is not in Open or Closed lists the search iteration will calculate and store fScore in Neighbour. And it assigns the reference of Parent Neighbour to Current node, then the current Neighbour is added to Open list.
- If Neighbour is already in Open list, then the new fScore will be calculated for the Neighbour factoring in current node.
- If new fScore is lower than fScore already stored in the Neighbour, then the current fScore will be replaced by the new fScore. Also, Parent reference will be updated for the current node.

III. UNITY 3D

Unity 3D is an incredibly powerful and versatile game and interactive-experience development tool created by Unity Technologies in 2005. Trendy game engines like unity's mecanim engine are unbelievably wealthy and powerful cross-platform. The core game engine includes a rendering engine for 2D and 3D graphics, a physics engine or collision detection, sound, scripting advanced animations, audio support, video support, and powerful UI-development tools. Any animation designed for Unity 3D must work with the logic of the sport loop. The Unity 3D game loop consists of a variable-time update and fixed-time update. Unity 3D scripting supports co-routines, that permits referred to as strategies, that permits referred to as strategies to possess multiple entries interleaved with the Unity 3D system. A package in Unity 3D may be an instrumentation of scenes and numerous assets like scripts, models, images, and sound effects employed by the sport objects inside those scenes.

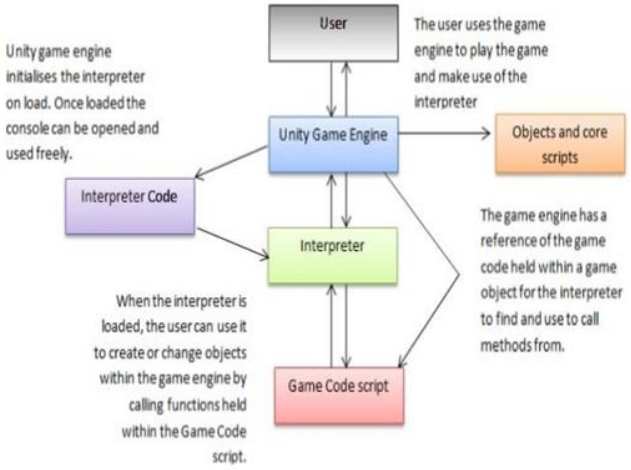


Fig.3: CSI interpreter design Different aspects of development process

- Rendering: Unity's graphics engine uses its own APIs like OpenGL, direct3D and also supports other file formats from other software's like Blender Adobe Photoshop etc.
- Scripting: Programmers write Unity Script similar to JavaScript on mono-an open source platform for .net framework.
- Physics: Provides built-in support for PhysX [10] physics engine with real time simulations on skinned meshes, thick ray casts etc.

IV. IMPLEMENTATION

A. STATE MACHINES

A device which can be in one of a set number of condition dependency on its previous condition and the present value of its input. It starts entry next node only if the given condition is true otherwise it will stay in their current node. Finite state machines are one of the most effective and most frequently used methods of programming artificial intelligence. Each state possesses code responsible for the initialization and de-initialization of the object. The finite state machine method lets us easily divide the implementation of each game object's behavior into smaller fragments. A long animation implemented in the update method may be controlled by a state machine to break the animation sequence into sequential pieces so that draws will not be blocked for long period of time.

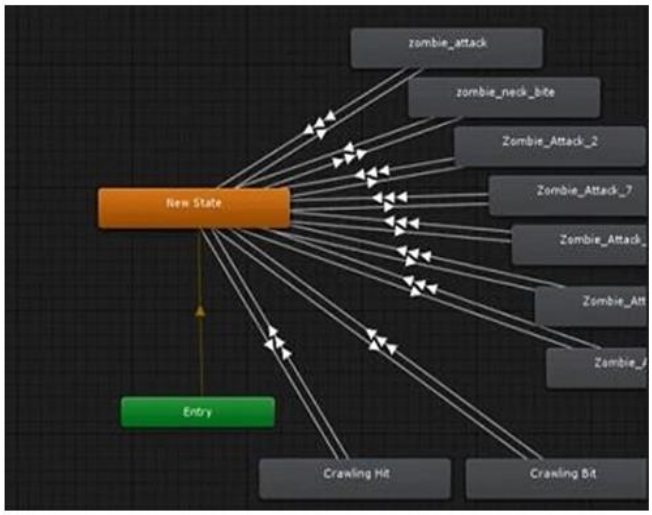


Fig.4: State machine

B. SCENE BUILDING, NAVIGATION AND PATH FINDING

A Scene contains the environments and menus of your game. Contemplate each distinctive Scene file as a singular level. In each Scene, you place your environments, obstacles, and decorations, essentially turning out with and building your game in things.

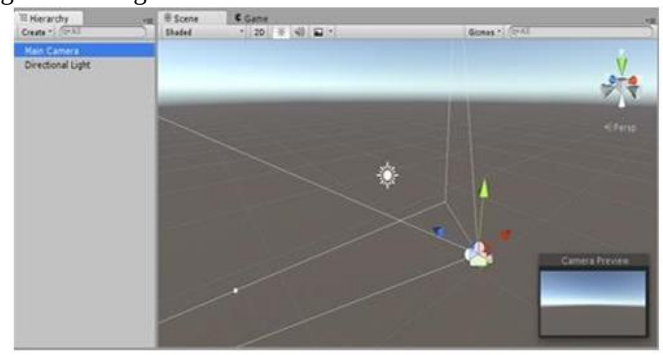


Fig.5: A new empty Scene, with the default 3D objects

a. Navigation mesh

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.

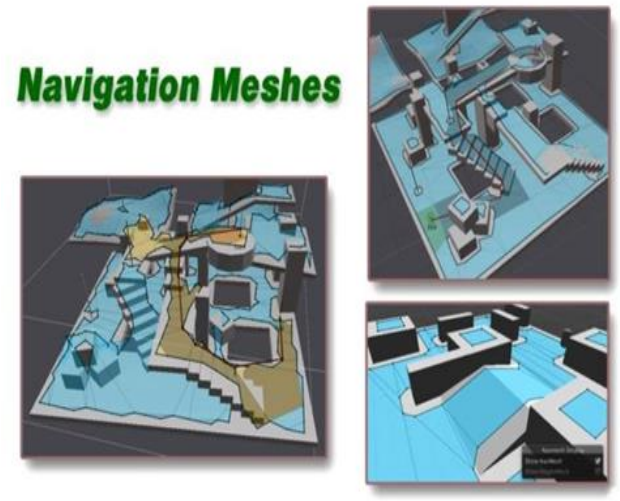


Fig.6: Game scene with NavMesh

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.

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.

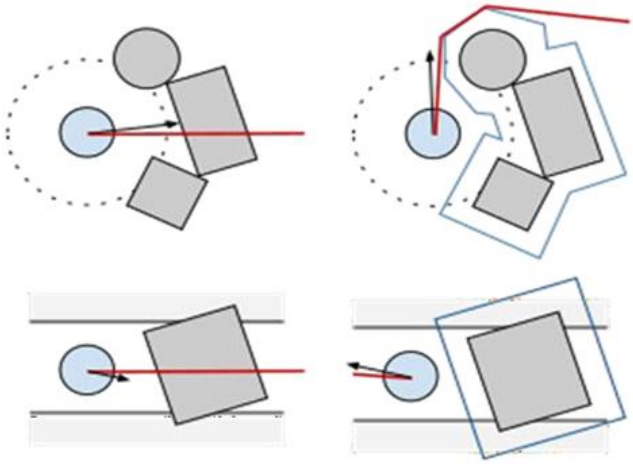


Fig.7: Avoiding obstacles in navigation

The controlling rationale takes the situation of the following corner and dependent on that makes sense of an ideal heading and speed (or speed) expected to achieve the goal. Utilizing the ideal speed to move the specialist can prompt impact with different operators.

Hindrance evasion picks another speed which adjusts between moving in the ideal bearing and avoiding future crashes with different specialists and edges of the route work. Solidarity is utilizing complementary speed deterrents (RVO) to anticipate and counteract impacts.

Numerous utilizations of navigation require different kinds of deterrents as opposed to simply different specialists. These could be the standard containers and barrels in a shooter amusement, or vehicles. The obstructions can be taken care of utilizing neighbourhood hindrance evasion or global pathfinding.

When associate obstacle is moving, it's best handled exploitation native obstacles shunning. this fashion the agent will predictively avoid the obstacle. once the obstacle becomes stationary, and might be thought-about to dam the trail of all agents, the obstacles ought to have an effect on the worldwide navigation, that is, the navigation mesh.

dynamic objects. Multiple agents and other dynamic objects also need to be able to happily co-exists. As the navigation graph is compiled at development time, path searches do not take into account objects which can moves at run time. A local avoidance system is employed to push dynamic objects always away from each other during agent steering. Unity can alter the navigation mesh at run time by carving objects into it. This is useful for when we know a dynamic object will never move again.

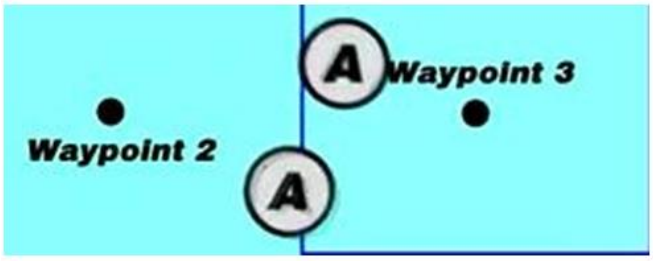


Fig.8: Local avoidance system

c. Path corridor

A path search in unity returns to the agent a corridor. A corridor is a list of polygons that must be traversed. Corridors are useful for supplying the agent with surrounding information so run-time path diversions can be safely computed. It is actually the vertices of the corridor that form the waypoint list the agent must pass through.

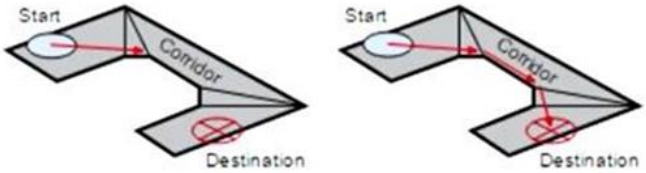


Fig.9: Run-time path diversions using Path corridor.

C. CHARACTER DESIGN

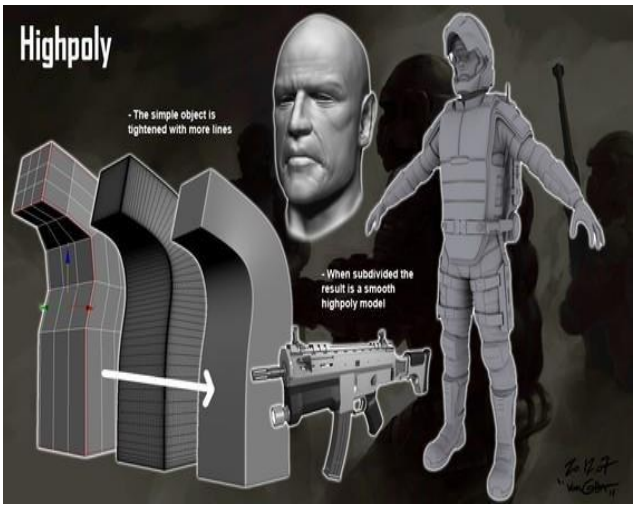
Changing the NavMesh is termed carving. the method detects that elements of the obstacle touches the NavMesh and carves holes into the NavMesh. this can be computationally overpriced operation, that is one more compelling reason, why moving obstacles ought to be handled exploitation collision shunning.

b. Local avoidance system

As other dynamic objects will not be represented in the navigation graph a separate system must exist that moderates the queried path to perform on-the-fly adjustment to the agent's velocity to avoid nearby dynamic objects. Only static objects are compiled into the navigation graph. Without the addition of a local avoidance system for dynamic objects,

agents may collide with each other or other When making a personality for a game there's typically a predefined work flow or pipeline which has plenty of steps. a number of the foremost common steps embody idea style, base modelling, high poly modelling, low poly modelling, unwrap, texturing and rigging/skinning. the primary step of making a personality is to come back up with an idea. an idea may be a painting or sketch, illustrating the specified look and theme of the finished character.

The next step is base modelling. during this step, the bottom model ought to include a rough model, with the proportions and shapes fairly correct whereas still keeping the pure mathematics straightforward. This step is sometimes wiped out a 3D special effects software package like liquidizer and Autodesk 3ds GHB. liquidizer could be a free and open supply software package used for modelling and making animated films, visual effects, interactive 3D applications or video games. once the bottom model has been created, a high poly model is to be created. this will be done by sculpting the



bottom model to feature additional detail to the character, for example wrinkles. thanks to the introduction of high level details, this step may be terribly long. The sculpting method may be delineated as shaping a coffee plane figure virtual clay mesh into a high polygon elaborated clay mesh.

Fig.10: A high polygon model

Rigging and Skinning

When the look and modelling of the character is complete, the character must be rigged and abraded so as to be able to use with animations within a game engine. Character animation is that the most advanced sort of animation. not like a tree, a personality not solely must be able to move, it conjointly should be able to categorical itself and show emotions. To rig the character, a skeleton designed along by completely different bones must be additional to the character. The bones must be connected to the character employing a skin modifier. This method is to make sure that the pure mathematics close the bones deforms properly once the skeleton moves. In liquidizer a skeleton might either be created by with either a separate bone affiliation approach or with Blender's equivalence of Biped, the Rigify system. once a skeleton has been created and placed so as to suit with the proportions and anatomy of the character, the bones should be abraded to the character. For skinning a Skin modifier is employed with this modifier, every bone can should be adjusted with envelopes and weights to attach the proper vertices to the bone for that specific part. rather than creating the character style method manually, there are tools to come up with a completely rigged and abraded low two- dimensional figure character with corresponding maps. this may provide artists an extremely high freedom of exploitation the creations from build Human in any manner imaginary.

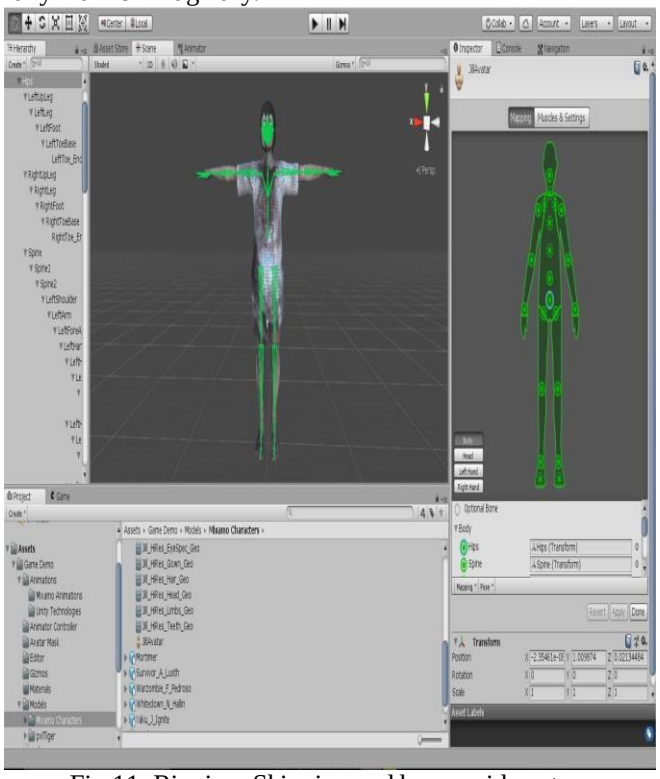


Fig.11: Rigging, Skinning and humanoid avatars

D. MECANIM

An important a part of 3D game development is animations, and additional specifically character animations. 2 of the best difficulties once desegregation the animations in a very game engine are generating diagrammatically swish transitions between different animations and additionally the advanced scripting that's needed for the various animation state transitions. However, Unity recently free Unity four. Unity four comes with the new animation system referred to as Mecanim.

Mecanim provides the user practicality to setup and management animations for a personality within the Unity editor. There are 2 styles of character varieties supported by Mecanim, particularly mechanical man and generic characters. A mechanical man may be a character that resembles the looks of a person's being whereas a generic character has no predefined bone structure, center of mass or orientation. If a generic character is to be used with Mecanim, a number of the elementary options provided by Mecanim won't be on the market. a straightforward example of a generic character may be a dog. once a mechanical man character is to be foreign to Unity and used with Mecanim, Mecanim creates Associate in Nursing Avatar. Associate in Nursing Avatar is an interface that interprets the characters bone structure to the bone structure understood by Mecanim. Fig.9 shows a translation mapping between the Mecanim bone structure and therefore

the actual bones within the character.

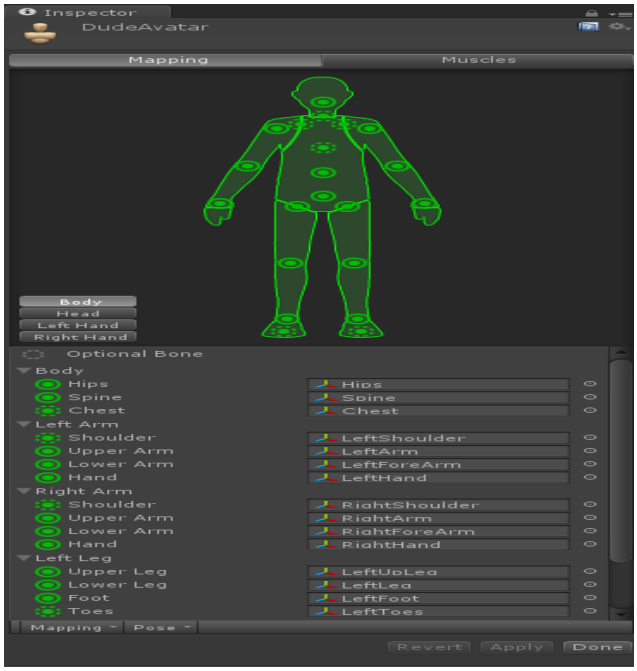


Fig.12: Avatar Configuration, mapping the character's bone structure to Mecanim's bone structure.

When a character has been successfully created with an Avatar, Mecanim allows the user to tell the game engine how and when different character animation clips should be played. A powerful tool within Mecanim is the Animator component.

E. ANIMATOR

In the Animator part, completely different states and layers are outlined and controlled. A state within the Animator part corresponds to the precise set of actions and therefore the surroundings of the character in the game. for instance, a state outlined as 'Idle' might correspond to a situation within the game wherever the character isn't moving in any respect. during this specific state, a whorled associate degree Imation clip of an idle motion may be applied to the state, forcing the character to play that animation whereas within the 'Idle' state. so as for the character to perform another action of movement, another state should be outlined. for instance, if the character is meant to run, a state known as 'Running' may be created, holding a clip of a whorled running motion. Also, transitions between the 2 states should be outlined, permitting the character to modify between the idle motion and a running motion. A transition is controlled by one or a lot of conditions that need to be met so as to modify between the states. A condition is controlled by a user outlined variable that successively is controlled by a script.

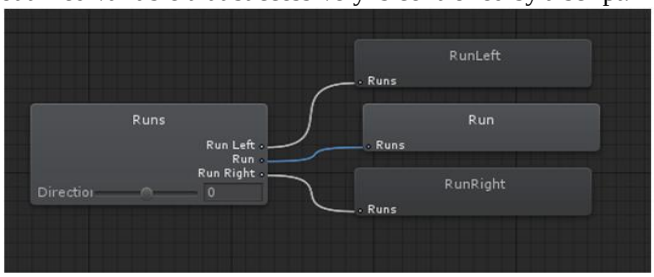


Fig.13: Animator component, showing an example of different animation states and the transitions between them.

Blend Tree

Another powerful feature of the Animator part is that the mix Trees. a mix Tree may be created within a state to interchange this animation clip therein state. for instance, the 'locomotion' state's walking motion clip may be replaced by a mix Tree, during which the recently created mix Tree might then embody a multiple variety of motion clips i.e., 'walk', and 'Run'. once the 'locomotion' state is active, the mix Tree plays one among its motion clips betting on a user outlined variable. during this example such a variable might represent the speed of the run so as to let the character walk or run consequently. If the worth of the speed variable is lesser than given threshold value it'll walk or if its larger than threshold value then it will begin running and also the animations will mix into a dynamic animation that interpolates between the 2 motions. This feature ends up in sleek and dynamic transition motions between completely different animation clips.

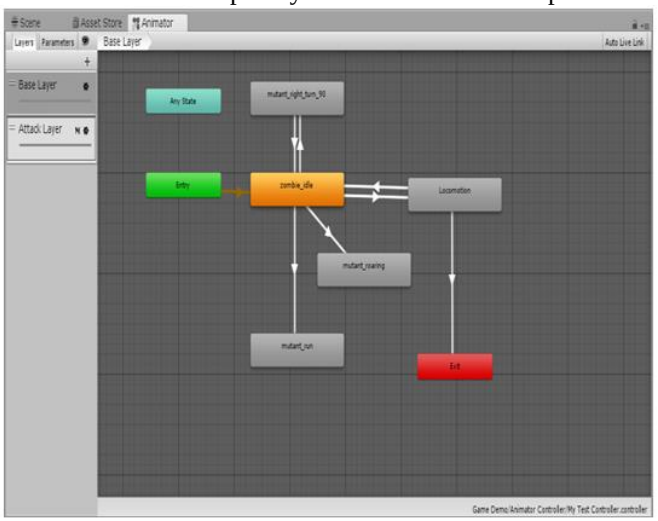


Fig.14: Blend Tree including three different animation clips.

V. CONCLUSION

In this paper, we have discussed various methods and algorithm that are being used in the field of first person shooter games. However, the combination of various evolutionary algorithm along with the finite state machine mechanisms have proven to be useful in generating AI bots for first person shooter.

First person shooters are new phase in gaming world. With their advanced technology and sophisticated environment, FP shooter gives a player thrilling experience. In every aspect it offers a great excitement with fast pace while playing. They give you the complete freedom to configure everything just the way you want it.

VI. REFERENCES

[1]. Geldenhuys, T. (2013) C# Interpreter Console for Unity 3D. [ONLINE] <http://blog.tiaan.comilink!2010/03115/csharpinterpreter-unity-plugin-console-debugger>.
[2]. Unity Asset Store: https://www.assetstore.unity3d.com.
[3]. Unity 3D: http://unity3d.com
[4]. S. M. LaValle, Planning algorithms, Cambridge University Press, 2006. [2] T. K. Whangbo, "Efficient Modified Bidirectional A * Algorithm for Optimal Route- Finding," New Trends in Applied Artificial Intelligence, Japan, vol.

4570, pp. 344–353, June 2007.
[5]. Y. Lang, "Research on collision detection method in unity," Software Guide, China, vol. 13, pp. 24–25, July 2014.
[6]. Kennedy J, Eberhart R C. Particle swarm optimization[A].In: Proceedings of IEEE International conference on Neural Networks[C] . Perth, Australia: [s. n.], 1995. 1942- 1948.
[7]. Fuellerer G, Doemer K F, Hartl R F, et al. Metaheuristics for vehicle routing problems with three- dimensional loading constraints[J]. European Journal of Operational Research, 2010, 201(3): 751-759.
[8]. McFarlane, A., Sparrowhawk, A. & Heald, y. 2013. Report on the educational use of games. [ONLINE] Available at: http://www . pewinternet.org/-/media/ /F ileslReportsl2008/PI P _Teens_ Games_and_ Civics_Report_FINAL. pdf. pdf. [Accessed 11 April 2013]. affiliations as succinct as possible (for example, do not differentiate among departments of the same organization).
[9]. Ismail Buyuksalih, Serdar Bayburt, Gurcan Buyuksalih, A P Baskaraca, Hairi KAlias AbdulRahman.#d modeling and visualization based on the Unity game engine-advantages and challenges.4th International GeoAdvance Workshop,2017.